TPC Benchmark™ A
Full Disclosure Report for

Silicon Graphics
CHALLENGE XL Server
and ORACLE7

April 26, 1994

SiliconGraphics, Inc.
Computer Systems

Document Number 007-2359-001

TPC Benchmark™ A Full Disclosure Report for Silicon Graphics Computer Systems CHALLENGE XL Server using ORACLE7.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ A test conducted by Silicon Graphics Computer Systems, with the assistance of ORACLE Corporation, on the Silicon Graphics Computer Systems CHALLENGE XL Server using ORACLE7. All tests were run on an Ethernet Local Area Network configuration with the CHALLENGE XL Server used as the host computer running the IRIX (UNIX) operating system. The application code was written in C and compiled with the IRIX-ANSI C compiler.

TPC Benchmark™ A Metrics

The standard TPC Benchmark™ A metrics, tpsA (transactions per second) and price per tpsA (five year capital cost per measured tpsA) are reported as required by the benchmark specification. Throughout this report, TPS refers to the tpsA performance metric. The next two pages contain the executive summaries of the benchmark results for the above systems.

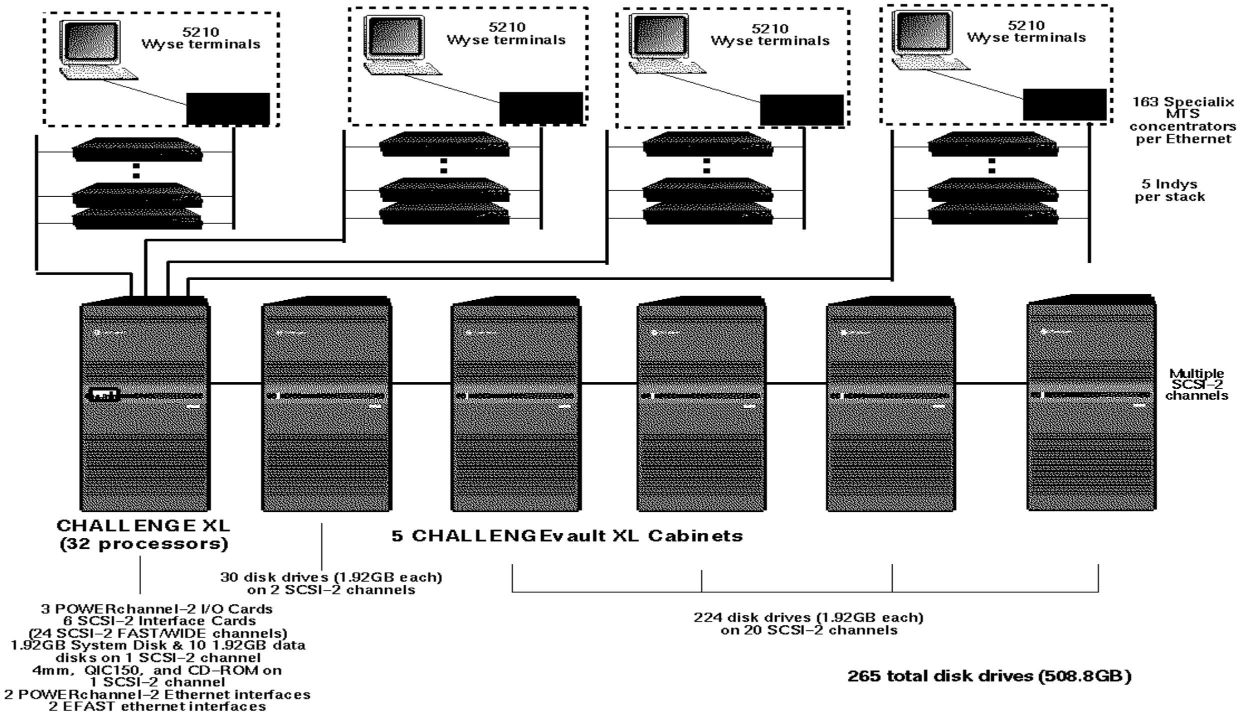
Auditor

The results of the benchmark tests, the methodology used to produce the results, and the calculations to produce the price per tpsA were independently audited by Performance Metrics, Inc. of Los Gatos, California.

Silicon Graphics Computer Systems	CHALLENGE XL Server c/s with 20 Indys	TPC-A Rev. 1.2 April 26, 1994

Total System Cost	TPC-A Throughput	Price / Performance
$11,137,661	2049.71 tpsA	$5,433.77 per tpsA

Processor	Database Manager	Operating System	Other Software	Number of Users
MIPS R4400SC	ORACLE7	IRIX 5.3	TUXEDO Release 4.2.1	20,840


5210 Wyse terminals
5210 Wyse terminals
5210 Wyse terminals
5210 Wyse terminals
163 Specialix MTS concentrators per Ethernet
5 Indys per stack
Multiple SCSI-2 channels
CHALLENGE XL (32 processors)
5 CHALLENGEvault XL Cabinets
30 disk drives (1.92GB each) on 2 SCSI-2 channels
3 POWERchannel-2 I/O Cards
6 SCSI-2 Interface Cards
(24 SCSI-2 FAST/WIDE channels)
1.92GB System Disk & 10 1.92GB data disks on 1 SCSI-2 channel
4mm, QIC150, and CD-ROM on 1 SCSI-2 channel
2 POWERchannel-2 Ethernet interfaces
2 EFAST ethernet interfaces
224 disk drives (1.92GB each) on 20 SCSI-2 channels
265 total disk drives (508.8GB)

System Components:	Qty:	Host and Development:	Qty:	Front-End Client:
Processors	32	MIPS R4400SC with 16KB I-cache, 16KB D-cache, and 4MB combined secondary cache	20	MIPS R4000SC with 8KB I-cache, 8KB D-cache and 1MB secondary cache
Memory	4	256MB (8-way interleaving)	20	256MB
I/O Controllers	3	POWERChannel-2 I/O controllers (2 each SCSI-2 FAST/WIDE channels)		
SCSI-2 Cards	6	3 FAST/WIDE SCSI-2 channels		
Disk Racks	5	CHALLENGEvault XL racks		
Total Disk Drives	265	Capacity - 508.8GB	20	1GB SCSI drive
Tape Drives	1	150MB QIC streaming tape		
	1	2GB DAT tape drive		
Terminals	1	System console	20,840	Terminals
Miscellaneous Peripherals	1	CD-ROM drive		
Terminal Concentrators			652	Terminal concentrators
Ethernet Interfaces	2	EFAST Ethernet interfaces	20	GIO Bus Ethernet cards
	2	POWERChannel-2 Ethernet Interfaces		

Silicon Graphics Computer Systems	CHALLENGE XL Server c/s with 20 Indys	TPC-A Rev. 1.2 April 26, 1994

Order Number	Description	Quantity	Unit Price	Extended Price	Support (5 years)
CHALLENGE XL Server					
R-45832-S4	32-CPU CHALLENGE XL Server, 4MB cache	1	$876,700	$876,700	$277,750
FTO-64UP512	First 512 MB High Dens Mem, 1 IMB	1	63,520	63,520	8,525
H4-512-4-ADD	Addt’l 512MB high Dens. Mem, 2 IMBs	1	82,800	82,800	13,975
P-S-B224	CHALLENGEvault XL 224GB Disk Bundle	2	560,000	1,120,000	184,800
P-S-B64	CHALLENGEvault XL 64GB Disk Bundle	1	196,250	196,250	26,400
P8-S-2	2GB SCSI-2 FAST/WIDE Disk	10	6,900	69,000	8,250
HU-PC2	POWERChannel-2 I/O Controller	2	12,000	24,000	8,500
P-S-HIO	SCSI-2 FAST/WIDE Interface Card	6	2,500	15,000	5,400
P8-QIC-CD	150 MB QIC tape & CD-ROM	1	2,000	2,000	1,700
P8-DAT	2GB DAT internal drive	1	2,500	2,500	1,125
P-TER2	110 VAC Programming Terminal	1	1,500	1,500	600
DK-C2-001	Destination Kit for XL Series	1	0.00	0.00	0.00
CC4-EFAST-2.0.1	Addt’l Ethernet Interface for CHALLENGE	2	5,700	11,400	2,250
DK-T2-001	Destination Kit for CHALLENGEvault XL	5	0.00	0.00	0.00
SC4-S4D-5.3	Operating System Software and Manuals	1	0.00	0.00	0.00
SC4-IDO-5.3	IRIX development options for IRIX 5.3	1	1,200	1,200	0.00
CS-SWCARE-DEV	Software options support (incl. IDO)	1	0.00	0.00	0.00
		TOTAL Server		**2,465,870**	**539,275**
Indy					
CH-S100	Indy, 100MHz, R4000SC, 1GB system disk	20	13,495	269,900	149,000
HU-M128A	128MB memory upgrade for Indy	40	18,000	720,000	0.00
CC2-E++-1.0	GIO Bus Ethernet Card	20	625	12,500	0.00
	32MB return to factory for Indy	20	-3,000	-60,000	0.00
		TOTAL Client		**942,400**	**149,000**
Communications & Terminals					
	†Specialix MTS	718	678	486,804	17,950
	†Specialix MTA 8-port expanders	2152	228	490,656	53,800
	Wyse WY-30+ terminals	20,840	189	3,938,760	729,400
	‡Anixter 8-port 10BaseT HubBNC	6	375	2,250	0.00
	‡Anixter Ethernet/IEEE Transceivers BNCTap	22	49	1,078	0.00
		TOTAL Comms.		**4,919,548**	**801,150**
ORACLE Software					
	ORACLE7 (448 named users)	1	358,400	358,400	215,040
	Procedural Option (448 named users)	1	71,680	71,680	43,008
	SQL*Net (448 named users)	1	71,680	71,680	43,008
	TCP/IP protocol driver (448 named users)	1	53,760	53,760	32,256
	SQL*Net (64 named users)	20	8,000	160,000	96,000
	TCP/IP protocol driver (64 named users)	20	6,000	120,000	72,000
Tuxedo Software					
	Tuxedo 4.2.1 (>10,001 users)	20,840	20	416,800	208,400
	Development system	1	380	380	190
		TOTAL Software		**1,252,700**	**709,902**
Discounts					
	Oracle Volume Discount			-150,393.60	-50,131.20
	Silicon Graphics CHALLENGE XL Discount			-394,539.20	0.00
	Silicon Graphics Indy Discount			-47,120.00	0.00
		TOTAL Discounts		**-592,052.80**	**-50,131.20**
Total Hardware and Software Costs					**11,137,661**
tps-A					**2049.71**
$/tps-A					**$5,433.77**

†includes 10% spares

‡ includes 10% spares and 5 year maintenance

Notes: Audited by Performance Metrics, Inc. of Los Gatos, CA.

Preface

Document Structure

Clause 10 of the TPC Benchmark™A specification describes the requirements for a full disclosure report. The main body of this document is organized as follows, based upon the requirements in Clause 10:

- Each portion of the main document begins with a Clause 10 requirement in an *italic* font. It is followed by normal font text that explains how each result complied with the requirement.

- Appendix A contains the source code of the application used to implement the benchmark.

- Appendix B describes the process that defines, creates, and loads the Oracle database. Also included are sample contents from each database table.

- Appendix C lists the tunable operating system and database parameters used in the benchmark test configuration.

- Appendix D contains the spreadsheet calculations used to determine the storage requirements for the ACCOUNT/-BRANCH/TELLER/HISTORY tables, eight (8) hour recovery log(s) and ninety (90) days of HISTORY.

- Appendix E contains the letter of attestation.

- Appendix F contains the source code for the RTE.
- Appendix G lists the Third Party Price Quotations.

TPC Benchmark™ A Overview

TPC Benchmark™ A was developed by the Transaction Processing Performance Council (TPC). It is the intent of the TPC to develop a suite of benchmarks to measure performance of computer systems across the spectrum of simple to complex applications.   Silicon Graphics Computer Systems is a member of the TPC.

TPC Benchmark™ A exercises the system components necessary to perform tasks associated with that class of online transaction processing environments emphasizing update-intensive database services. Such environments are characterized by:

- Multiple online terminal sessions
- Significant disk input/output
- Moderate system and application execution time
- Transaction integrity

This benchmark uses a single, simple update-intensive transaction to load the system under test (SUT). Thus, the workload is intended to reflect an OLTP application, but does not reflect the entire range of OLTP requirements typically characterized by multiple transactions types of varying complexities. The single transaction type provides a simple, repeatable unit of work, and was designed to exercise the key components of an OLTP system.

The metrics reported in TPC Benchmark™ A are throughput as measured in transactions per second, subject to a residence time constraint, and the associated price-per-tps. The throughput metrics are “tpsA”.

The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely the customer's application approximates TPC Benchmark™ A. Relative system performance of systems derived from TPC Benchmark™ A do not necessarily hold for other workloads or environments. Extrapolations to dissimilar environments are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary because of these and other factors. Therefore, TPC Benchmark™ A should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

All performance data contained in this report was obtained in a rigorously controlled environment, and therefore results obtained in other operating environments may vary significantly. Silicon Graphics Computer Systems does not warrant or represent that a user can or will achieve similar performance expressed in transactions per second (tpsA) or normalized price/performance ($/tpsA). No warranty of system performance or price/performance is expressed or implied in this report.

Table of Contents

Abstract iii
Overview iii
TPC Benchmark™ A Metrics iii
Auditor iii

Preface vii
Document Structure vii
TPC Benchmark™ A Overview viii

Clause 2 Transaction System Properties 2-1
2.1 Transaction System Properties (ACID) 2-1
2.2 Atomicity 2-1
2.2.1 Completed Transaction 2-2
2.2.2 Aborted Transaction 2-3
2.3 Consistency 2-4
2.4 Isolation 2-5
2.4.1 Completed Transaction 2-6
2.4.2 Aborted Transaction 2-7
2.5 Durability 2-7
2.5.1 Permanent Irrecoverable Failure 2-8
2.5.2 Instantaneous Interruption 2-9
2.5.3 Loss of Memory 2-9

Clause 3 Logical Database Design 3-1
3.1 Database Design 3-1
3.1.1 Distribution and Partitioning 3-1
3.1.2 Population and Sample Contents 3-2
3.1.3 Type of Database 3-2

Clause 4 Scaling Rules 4-1
4.1 Clause 4 Related Items 4-1
4.1.1 Database Scaling, and Row Occurrences 4-1

Clause 5 Distribution, Partitioning, and Transaction Generation 5-1
5.1 Random Number Generator 5-1
5.2 Horizontal Partitioning 5-2

Clause 6 Response Time 6-1
6.1 Benchmark Performance 6-1
6.1.1 Response Time 6-1
6.1.2 Throughput (tpsA) vs. Response Time 6-2

Clause 7 Duration of Test 7-1
7.1 Steady State 7-1
7.2 Work Performed During Steady State 7-1
7.3 Reproducibility 7-2
7.4 Measurement Period Duration 7-3

Clause 8 SUT & Driver 8-1
8.1 RTE 8-1

8.2 Driver Functionality and Performance 8-4
8.3 Network Bandwidth 8-6
8.4 Think Time 8-6

Clause 9 Pricing 9-1
9.1 System Pricing 9-1
9.1.1 CHALLENGE XL Server 9-1
9.1.2 Support Pricing 9-2
9.1.3 Priced System Configuration 9-2
9.1.4 Discounts 9-2
9.2 Availability 9-2
9.3 Measured tpsA 9-3
9.4 Priced Storage Requirements 9-3

Clause 10 Full Disclosure Checklist 10-1
10.1 General Items 10-1
10.2 Clause 2 Related Items 10-2
10.3 Clause 3 Related Items 10-2
10.4 Clause 5 Related Items 10-3
10.5 Clause 6 Related Items 10-3
10.6 Clause 7 Related Items 10-3
10.7 Clause 8 Related Items 10-4
10.8 Clause 9 Related Items 10-5
10.9 Clause 11 Related Items 10-6
10.10 FDR Availability 10-6

Clause 11 Related Items 11-1
11.1 Independent Auditing 11-1

Appendix A: Application Source Code A-1
Client Application A-1
Tuxedo Server A-6

Appendix B: Database Definition and Load B-1
File Definitions for ABTH Tables B-1
ABTH Table Sample Data B-15

Appendix C: Tunable Parameters C-1
Indy Operating System Tunable Parameters C-1
Challenge XL Operating System Tunable Parameters C-1
ORACLE7 Configuration C-1

Appendix D: Storage Requirements D-1
Disk Storage Requirements D-1

Appendix E: Attestation Letter E-1

Appendix F: RTE F-1
RTE F-1

Appendix G: Third-Party Vendor Quotations G-1

Clause 2 Transaction System Properties

2.1 Transaction System Properties (ACID)

Results of the ACIDity test (specified in Clause 2) must describe how the requirements were met. If a database different from that which is measured is used for durability tests, the sponsor must include a statement that durability works on the fully loaded and fully scaled database.

The TPC Benchmark™ A Standard Specification defines a set of transaction processing system properties that a System Under Test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation and Durability (ACID). This portion of the document will define each of those properties and describe the series of tests that were performed to demonstrate that the properties were met.

All of the specified ACID tests were performed on the CHALLENGE XL Server. Each ACID test was performed on the measured database.

The test to fail a single durable medium with table/file data was run on the database scaled to two thousand two hundred (2200) tpsA.

2.2 Atomicity

The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations have any effects on the data.

The following tests for atomicity were successfully completed for both regular transactions and discrete transactions.

2.2.1 Completed Transaction

Perform the standard TPC Benchmark™ A transaction (see Clause 1.2) for a randomly selected account and verify that the appropriate records have been changed in the Account, Branch, Teller, and History files/tables.

A verification of a commited transaction was completed as follows:

- a random Account and Teller were selected
- the current balances for the selected Account, Teller, and the Teller's associated Branch were recorded
- the number of rows in the History table that contain the above combination of Account, Branch, and Teller was recorded
- an automated version of the TPC Benchmark™ A application was executed that prompts a terminal for the transaction input and allows the user the option of COMMITting or ABORTing the transaction.
- the selected Account and Teller identifiers along with a random delta amount was entered for the transaction,
- the TPC Benchmark™ A application updated the appropriate Account, Branch, and Teller balances with the above delta amount, inserted an appropriate entry in the History table and prompted the user to either COMMIT or ABORT the current transaction,
- a COMMIT request was issued from the terminal
- the TPC Benchmark™ A application COMMITted the above transaction as requested.

After the transaction was COMMITted:

- the balances from the selected Account, Branch, and Teller were displayed
- it was verified that the displayed balances differed from the original balances by the delta value that was entered,
- the number of rows in the History table for the combination of the selected Account, Branch, and Teller was displayed

- it was verified that the number of History table rows was one greater than before the above transaction was executed,

- it was verified that the additional History row contained the proper values from the transaction entered.

2.2.2 Aborted Transaction

Perform the standard TPC Benchmark™ A Transaction for a randomly selected account, substituting an ABORT of the transaction for the COMMIT of the transaction. Verify that the appropriate records have not been changed in the Account, Branch, Teller, and History files/tables.

A verification of an aborted transaction was completed as follows:

- a random Account and Teller were selected

- the current balances for the selected Account, Teller, and the Teller's associated Branch were recorded

- the number of rows in the History table that contain the above combination of Account, Branch, and Teller was recorded

- an interactive version of the TPC Benchmark™ A application was executed that prompts a terminal for the transaction input and allows the user the option of COMMITting or ABORTing the transaction,

- the selected Account and Teller identifiers along with a random delta amount was entered for the transaction,

- the TPC Benchmark™ A application updated the appropriate Account, Branch, and Teller balances with the above delta amount, inserted an appropriate entry in the History table and prompted the user to either COMMIT or ABORT the current transaction,

- an ABORT request was issued from the terminal

- the TPC Benchmark™ A application ABORTed the above transaction as requested.

After the transaction was ABORTed:

- the balances from the selected Account, Branch, and Teller were displayed

- it was verified that the displayed balances were the same as before the transaction was started

- the number of rows in the History table for the combination of the selected Account, Branch, and Teller was displayed

- it was verified that the number of History table rows was no different than before the above transaction was executed,

2.3 Consistency

Consistency is the property of the application that requires any execution of a transaction to take the database from one consistent state to another.

A consistent state for the TPC Benchmark™ A database is defined to exist when:

a) the sum of the account balances is equal to the sum of the teller balances, which is equal to the sum of the branch balances;

b) for all branches, the sum of the teller balances within a branch is equal to the branch balance;

c) the history file has one logical record added for each committed transaction, none for any aborted transaction, and the sum of the deltas in the records added to the history file equals the sum of the deltas for all committed transactions.

If data is replicated, each copy must not violate these conditions.

Due to the large size of the Account file/table, no test of its consistency is specified.

The following tests were performed on the system under test (SUT) to demonstrate the property of consistency.

Prior to executing the TPC Benchmark™ A transactions:

- the balance for each Branch occurrence in the database was recorded (Initial Branch Balances),

- the sum of the above balances of all the Branches were recorded (Initial Branch Sum),

- the sum of the Teller balances within each branch were recorded (Initial Teller/Branch Balance),

- it was verified that the Initial Branch Balance equaled the sum of the Initial Teller/Branch Balances for each Branch,

- the number of History rows and the sum of the History delta values were recorded (Initial History Count and Initial History Sum),

- the TPC Benchmark™ A applications was executed and the number of committed transactions was recorded. It was verified that the number of committed transactions was not less than ten (10) times the number of Teller occurrences.

After the TPC Benchmark™ A application was executed:

- the sum of the balances of all Branch occurrences in the database was recorded (Final Branch Sum),

- the balance for each Branch occurrence in the database was recorded (Final Branch Balances),

- for each Branch, the sum of Teller balances associated with the Branch was recorded (Final Teller/Branch Balance),

- for each Branch, it was verified that the Final Branch Balance equaled the appropriate Final Teller/Branch Balance,

- the number of History rows and the sum of History row delta values amounts were recorded (Final History Count and Final History Sum),

- it was verified that the difference between the Final History Count and Initial History Count was the number of transactions recorded as committed,

- it was verified that the difference between the Final History Sum and Initial History Sum equaled the difference between the Final Branch Sum and Initial Branch Sum.

The benchmark was run and the appropriate number of transactions was executed and committed. In each of the above cases, the appropriate values and relationships were observed. The sum of Teller balances associated with a particular Branch equaled that Branch's balance before and after the execution of the benchmark. The difference between the Final and Initial History Counts was equal to the number of recorded committed transactions. The difference between the Final and Initial History Sum equaled the difference between the Final and Initial Branch Sum.

2.4 Isolation

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

This property is commonly called serializability. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any mix of arbitrary transactions, not just TPC Benchmark™ A transactions. The system or application must have full serializability enabled, i.e., repeated reads of the same records within any committed transactions must have returned identical data when run concurrently with any mix of arbitrary transactions.

Twenty four (24) Isolation tests were performed for both discrete and normal combinations, both COMMITted and ABORTed transactions for the Branch, Account, and Teller tables. The following two tables show the steps used in performing the Isolation test for the Account table with a COMMITted transaction (Table 2.1) and an ABORTed transaction (Table 2.2). The same steps were used to test both the Branch and Teller tables.

2.4.1 Completed Transaction

Table 2.1: Isolation Test — Completed Transaction

Transaction 1	Transaction 2
Execute a TPC Benchmark™ A transaction to update a randomly selected Account, using the application code described in the Atomicity tests. Stop the transaction prior to COMMIT.	
	Execute a second TPC Benchmark™ A transaction that will update the same Account as Transaction 1 using a different Teller and Branch. This transaction will wait until Transaction 1 completes.
COMMIT this transaction and verify the Account balance reflects the effect of the update.	
	This transaction resumes and is COMMITted. The Account balance reflects the effect of both Transaction 1 and Transaction 2.

The aborted transaction tests (Table 2.2) follow on the next page.

2.4.2 Aborted Transaction

Table 2.2: Isolation Test — Aborted Transaction

Transaction 1	Transaction 2
Execute a TPC Benchmark™ A transaction to update a randomly selected Account, using the application code described in the Atomicity tests. Stop the transaction prior to COMMIT.	
	Execute a second TPC Benchmark™ A transaction that will update the same Account as Transaction 1 using a different Teller and Branch. This transaction will wait until Transaction 1 completes.
ABORT this transaction and verify the Account balance remains unchanged.	
	This transaction resumes and is COMMITted. The Account balance reflects only the effect of Transaction 2.

2.5 Durability

The tested system must guarantee the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed below:

- Permanent irrecoverable failure of any single durable medium containing database, ABTH files/tables, or recovery log data.
- Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.
- Failure of all or part of memory (loss of contents).

A durable medium is a data storage medium that is either:

a) *an inherently non-volatile medium, e.g., magnetic disk, magnetic tape, optical disk, etc., or*

a) *a volatile medium with its own self-contained power supply that will retain and permit the transfer of data, before any data is lost, to an inherently non-volatile medium after the* *failure of external power.*

A transaction is considered committed when the transaction manager component of the system has written the commit record(s) associated with the transaction to a durable medium.

It is required that the system crash test and the loss of memory test described in Clauses 2.5.3.2 and 2.5.3.3, respectively, be performed with a full terminal load and a fully scaled database. The durable media failure tests described in Clause 2.5.3.1 may be performed on a subset of the SUT configuration and database. For that subset, all multiple hardware components, such as processors and disk/controllers in the full configuration must be represented by either 10% or 2 each of the multiple hardware components, whichever is greater. The database subset must be scaled to at least 10% (minimum of 2 tps) of the fully scaled database size. The test sponsor must state that to the best of their knowledge, a fully loaded and fully scaled SUT and database configuration would also pass all durability tests.

At the time of the induced failures, it is required to have multiple home and remote transactions (see Clause5) in progress. Distributed configurations must have distributed transactions in progress as well.

All of the Durability tests listed below completed successfully. The sum of Teller balances associated with a particular Branch equaled that Branch's balance before and after the execution of the benchmark. The difference between the Final and Initial History Counts was equal to the number of recorded committed transactions. The difference between the Final and Initial History Sum equaled the difference between the Final and Initial Branch Sum, and every record in the 'success' file had a corresponding row occurrence in the History tables.

The failures listed below were induced on the system under test (SUT) to demonstrate the property of Durability.

2.5.1 Permanent Irrecoverable Failure

Permanent irrecoverable failure of any single durable medium containing database, ABTH files/tables, or recovery log data.

Two irrecoverable failures were tested, one for failure of table and catalog medium, and another for database recovery log medium.

The table and catalog medium failure was tested as follows:

- a failure was induced by copying bad data over the sections of the disk that stored the database catalog and another disk containing account data. This caused appropriate error messages to appear on the console and the application to stop.

- the database was shut down,
- the backup was restored, overwriting the existing contents of the disk, and the database was rolled forward using the recovery log file,
- the count of records in the success file was compared to the rows in the History table to verify that all transactions were correctly recovered,
- random rows from the success file were searched out in the History table to verify the contents were successfully recovered

The recovery log medium was mirrored. Failure of the recovery log was tested as follows:

- while transactions were being processed, one of the mirrored disks was physically removed from the SUT,
- processing continued unaffected and no recovery was necessary

2.5.2 Instantaneous Interruption

Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.

See section 2.5.3.

2.5.3 Loss of Memory

Failure of all or part of memory (loss of contents). Perform the consistency test on the Branch and Teller files as specified in Clause 2.3.3.2.

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory.

- a consistency check was run and the file system was synchronized to ensure the audit files were written to disk and would not be lost,
- while transactions were being processed, a failure was induced by turning off the primary power for the SUT,
- power to the SUT was restored and ORACLE7 was restarted,
- the database was rolled forward using the recovery log file,

- the sum of all branch balances was compared to the sum of all teller balances and verified to be the same,

- for each branch, the sum of the balances for all local tellers was compared to the stored balance for the branch and verified to be the same,

- the count of records in the success file was compared to the rows in the History table to verify that all committed transactions were correctly recovered.

Clause 3 Logical Database Design

3.1 Database Design

The distribution across storage media of ABTH (Accounts, Branch, Teller, and History) files/tables and all logs must be explicitly depicted.

3.1.1 Distribution and Partitioning

This benchmark was implemented as a centralized solution accessing a single logical and physical database. All tables and the History file were partitioned horizontally across multiple disk drives. That horizontal partitioning was transparent to the TPC Benchmark™ A application. Vertical partitioning was not used in this benchmark implementation.

The benchmark and priced system configuration diagrams are shown in Figures 3.1 and 3.2, respectively. The SUT utilized 202 disk drives in its configuration. The test was executed against a database built for 2200 branches of this database.

The specific distribution and partitioning across storage media of database tables (Account, Branch, Teller, and History) and recovery logs, are graphically depicted for both the benchmark and the priced configuration in Table 3.1 and Table 3.2 respectively. The same allocations were used for both the benchmark and the priced configuration. The only difference is the amount of data generated for History and Log files during the benchmark did not completely fill all allocated blocks.

The ACCOUNT file was equally allocated across 160 disks.

The BRANCH file was allocated on 1 disk.

The TELLER file was allocated on 1 disk.

The HISTORY file was equally allocated across 13 disks, 1 extent each.

3.1.2 Population and Sample Contents

A description of how the database was populated, along with sample contents of each ABTH file/table to meet the requirements described in Clause 3.

Appendix B describes the process that defines, creates/installs, and populates the ORACLE7 on-line database for TPC Benchmark™ A. Sample contents of each database table are included in this appendix.

3.1.3 Type of Database

A statement of the type of database utilized, e.g., relational, Codasyl, flat file, etc.

This TPC Benchmark™ A used the ORACLE7 RDBMS relational database software.

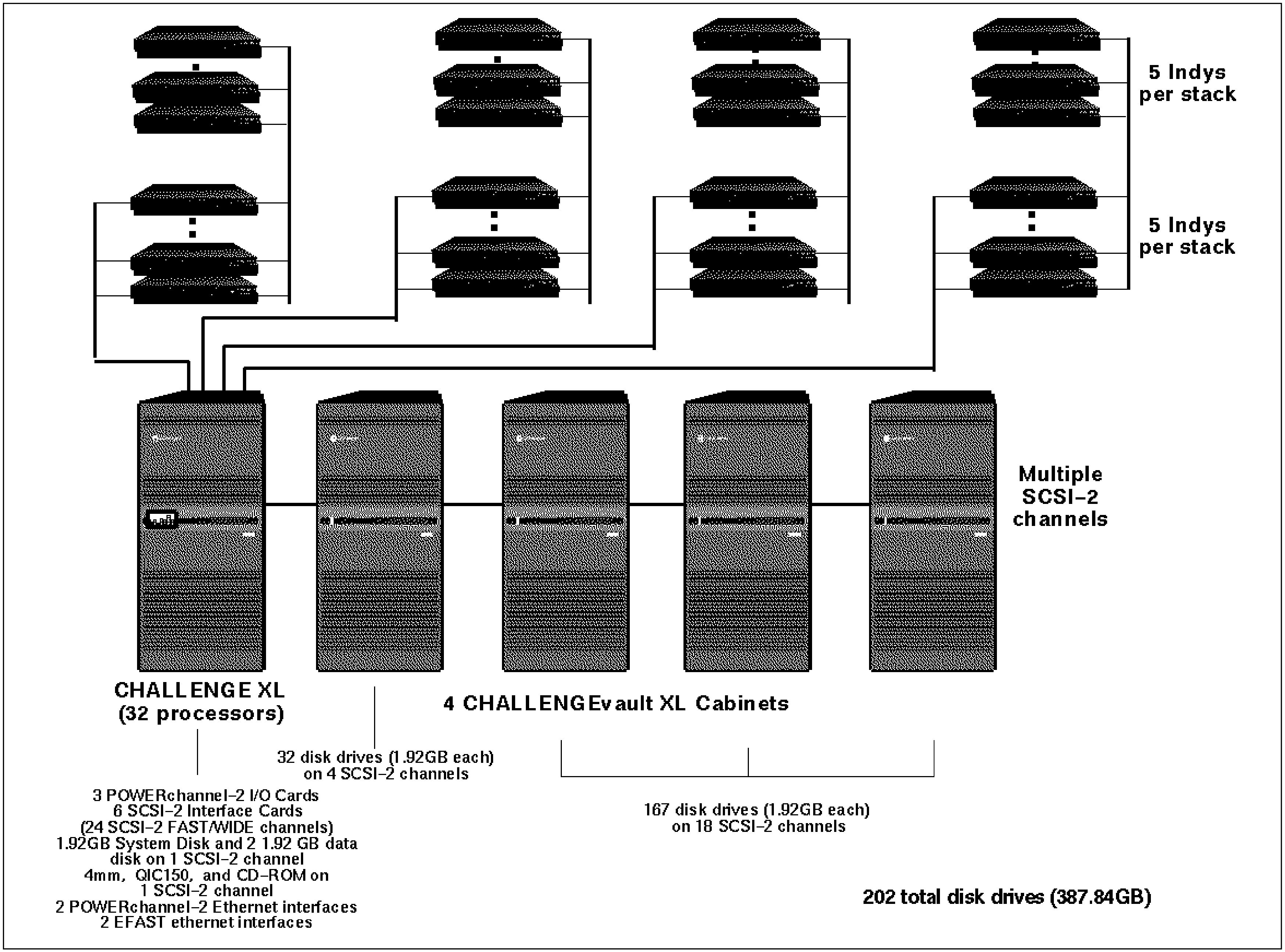

5 Indys per stack
5 Indys per stack
Multiple SCSI-2 channels
CHALLENGE XL (32 processors)
4 CHALLENGEvault XL Cabinets
32 disk drives (1.92GB each) on 4 SCSI-2 channels
3 POWERchannel-2 I/O Cards
6 SCSI-2 Interface Cards
(24 SCSI-2 FAST/WIDE channels)
1.92GB System Disk and 2 1.92 GB data disk on 1 SCSI-2 channel
4mm, QIC150, and CD-ROM on 1 SCSI-2 channel
2 POWERchannel-2 Ethernet interfaces
2 EFAST ethernet interfaces
167 disk drives (1.92GB each) on 18 SCSI-2 channels
202 total disk drives (387.84GB)

Figure 3.1: Benchmark System Configuration

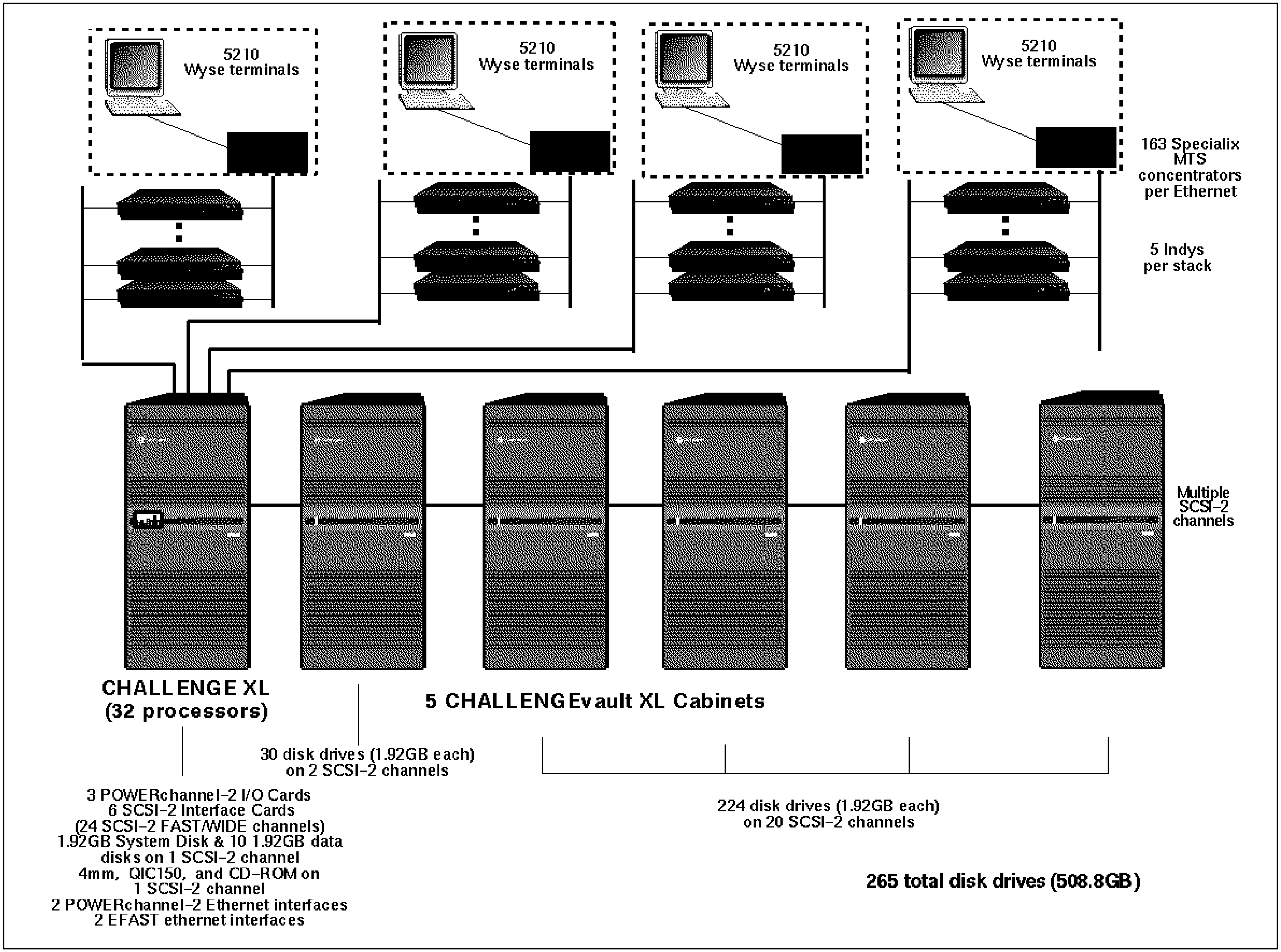

5210 Wyse terminals
5210 Wyse terminals
5210 Wyse terminals
5210 Wyse terminals
163 Specialix MTS concentrators per Ethernet
5 Indys per stack
Multiple SCSI-2 channels
CHALLENGE XL (32 processors)
5 CHALLENGEvault XL Cabinets
30 disk drives (1.92GB each) on 2 SCSI-2 channels
3 POWERchannel-2 I/O Cards
6 SCSI-2 Interface Cards
(24 SCSI-2 FAST/WIDE channels)
1.92GB System Disk & 10 1.92GB data disks on 1 SCSI-2 channel
4mm, QIC150, and CD-ROM on 1 SCSI-2 channel
2 POWERchannel-2 Ethernet interfaces
2 EFAST ethernet interfaces
224 disk drives (1.92GB each) on 20 SCSI-2 channels
265 total disk drives (508.8GB)

Figure 3.2: Priced System Configuration

Disk	Total	UNIX	ORACLE	Branch	Account	Account	History	Log	Log
Name	# of	+swap	System&	&Teller	data	Index	Data	Data	Data
		Drives		Control files	data				(Mirror)
				% of data/disk					
dks110d[1-8]	8	0	0	0	.625	0	0	0	0
dks111d[1-8]	8	0	0	0	.625	0	0	0	0
dks113d[1-8]	8	0	0	0	.625	0	0	0	0
dks114d[1-8]	8	0	0	0	.625	0	0	0	0
dks115d[1-8]	8	0	0	0	.625	0	0	0	0
dks116d[1-2]	2	0	0	0	.625	20	0	0	0
dks116d[3-8]	6	0	0	0	.625	0	0	0	0
dks116d9	1	0	0	0	0	0	0	7.70	0
dks116d10	1	0	0	0	0	0	0	7.70	0
dks116d1[1-2]	2	0	0	0	0	0	0	0	7.70
dks117d[1-2]	2	0	0	0	.625	20	0	0	0
dks117d[3-8]	6	0	0	0	.625	0	0	0	0
dks117d9	1	0	0	0	0	0	0	7.70	0
dks117d10	1	0	0	0	0	0	0	7.70	0
dks117d1[1-2]	2	0	0	0	0	0	0	0	7.70
dks1d1	1	100	0	0	0	0	0	0	0
dks1d[2-3]	2	0	0	0	.625	0	0	0	0
dks2d1	1	0	0	0	.625	0	0	0	0
dks2d2	1	0	0	0	.625	20	0	0	0
dks2d3	1	0	0	0	0	0	8.60	0	0
dks2d[4-7]	4	0	0	0	.625	0	0	0	0
dks2d8	1	0	0	0	0	0	8.60	0	0
dks3d1	1	0	0	0	.625	0	0	0	0
dks3d[2-7]	6	0	0	0	.625	0	0	0	0
dks3d8	1	0	100	0	0	0	0	0	0
dks3d9	1	0	0	0	0	0	8.60	0	0
dks3d10	1	0	0	0	0	0	4.30	0	0
dks3d11	1	0	0	0	0	0	0	7.70	0
dks3d12	1	0	0	0	0	0	0	0	7.70
dks4d1	1	0	0	0	.625	0	0	0	0
dks4d[2-7]	6	0	0	0	.625	0	0	0	0
dks4d8	1	0	0	0	0	0	4.30	0	0
dks4d9	1	0	0	0	0	0	0	7.70	0
dks4d10	1	0	0	0	0	0	0	0	7.70
dks4d11	1	0	0	0	0	0	0	7.70	0
dks4d12	1	0	0	0	0	0	0	0	7.70
dks5d[1-7]	7	0	0	0	.625	0	0	0	0
dks5d8	1	0	0	0	0	0	8.60	0	0
dks5d9	1	0	0	0	0	0	4.30	0	0
dks5d10	1	0	0	0	0	0	8.60	0	0
dks5d11	1	0	0	0	0	0	9.74	0	0
dks5d12	1	0	0	0	0	0	8.60	0	0
dks6d[1-7]	7	0	0	0	.625	0	0	0	0
dks6d8	1	0	0	0	0	0	8.60	0	0
dks6d9	1	0	0	0	0	0	0	7.70	0

Table 3.1: SUT Configuration Data Distribution

Disk	Total	UNIX	ORACLE	Branch	Account	Account	History	Log	Log
Name	# of	+swap	System&	&Teller	data	Index	Data	Data	Data
		Drives		Control files	data				(Mirror)
		→		% of data/disk		←			
dks6d10	1	0	0	0	0	0	0	0	7.70
dks6d11	1	0	0	0	0	0	0	7.70	0
dks6d12	1	0	0	0	0	0	0	0	7.70
dks70d[1-8]	8	0	0	0	.625	0	0	0	0
dks71d[1-7]	7	0	0	0	.625	0	0	0	0
dks71d8	1	0	0	0	0	0	0	7.70	0
dks71d9	1	0	0	0	0	0	0	0	7.70
dks71d10	1	0	0	0	0	0	0	7.70	0
dks71d11	1	0	0	0	0	0	0	0	7.70
dks71d12	1	0	0	0	0	0	0	7.70	0
dks71d13	1	0	0	0	0	0	0	0	7.70
dks71d1[4-5]	2	0	0	0	0	0	8.60	0	0
dks72d[1-7]	7	0	0	0	.625	0	0	0	0
dks72d8	1	0	0	0	0	0	0	0	7.70
dks73d[1-7]	7	0	0	0	.625	0	0	0	0
dks73d8	1	0	0	100	0	0	0	0	0
dks74d[1-8]	8	0	0	0	.625	0	0	0	0
dks75d[1-8]	8	0	0	0	.625	0	0	0	0
dks76d[1-8]	8	0	0	0	.625	0	0	0	0
dks77d[1-8]	8	0	0	0	.625	0	0	0	0
dks7d[1-7]	7	0	0	0	.625	0	0	0	0
dks7d8	1	0	0	0	0	0	0	7.70	0

Table 3.1 SUT Configuration Data Distribution (continued)

Disk Name	Total # of	UNIX +swap Drives	ORACLE System&	Branch &Teller Control files	Account data data	Account Index	History Data	Log Data	Log Data (Mirror)
					% of data/disk				
dks110d[1-8]	8	0	0	0	.625	0	.4	0	0
dks110d9	1	0	0	0	0	0	.485	0	0
dks110d1[0-2]	3	0	0	0	0	0	.485	0	0
dks111d[1-8]	8	0	0	0	.625	0	.4	0	0
dks111d9	1	0	0	0	0	0	.485	0	0
dks111d1[0-2]	3	0	0	0	0	0	.485	0	0
dks112d[1-9]	9	0	0	0	0	0	.485	0	0
dks112d1[0-5]	6	0	0	0	0	0	.485	0	0
dks113d[1-8]	8	0	0	0	.625	0	.4	0	0
dks113d9	1	0	0	0	0	0	.485	0	0
dks113d1[0-2]	3	0	0	0	0	0	.485	0	0
dks114d[1-8]	8	0	0	0	.625	0	.4	0	0
dks114d9	1	0	0	0	0	0	.485	0	0
dks114d1[0-2]	3	0	0	0	0	0	.485	0	0
dks115d[1-8]	8	0	0	0	.625	0	.4	0	0
dks115d9	1	0	0	0	0	0	.485	0	0
dks115d1[0-2]	3	0	0	0	0	0	.485	0	0
dks116d[1-2]	2	0	0	0	.625	20.00	.173	0	0
dks116d[3-8]	6	0	0	0	.625	0	.4	0	0
dks116d9	1	0	0	0	0	0	0	7.70	0
dks116d10	1	0	0	0	0	0	0	7.70	0
dks116d1[1-2]	2	0	0	0	0	0	0	0	7.70
dks117d[1-2]	2	0	0	0	.625	20.00	.173	0	0
dks117d[3-8]	6	0	0	0	.625	0	.4	0	0
dks117d9	1	0	0	0	0	0	0	7.70	0
dks117d10	1	0	0	0	0	0	0	7.70	0
dks117d1[1-2]	2	0	0	0	0	0	0	0	7.70
dks1d1	1	100.00	0	0	0	0	.275	0	0
dks1d[2-3]	2	0	0	0	.625	0	.4	0	0
dks1d[4-9]	6	0	0	0	0	0	.485	0	0
dks1d1[0-1]	2	0	0	0	0	0	.485	0	0
dks2d1	1	0	0	0	.625	0	.4	0	0
dks2d2	1	0	0	0	.625	20.00	.173	0	0
dks2d3	1	0	0	0	0	0	.485	0	0
dks2d[4-7]	4	0	0	0	.625	0	.4	0	0
dks2d[8-9]	2	0	0	0	0	0	.485	0	0
dks2d1[0-2]	3	0	0	0	0	0	.485	0	0
dks3d1	1	0	0	0	.625	0	.4	0	0
dks3d[2-7]	6	0	0	0	.625	0	.4	0	0
dks3d8	1	0	100.00	0	0	0	0	0	0
dks3d9	1	0	0	0	0	0	.485	0	0
dks3d10	1	0	0	0	0	0	.485	0	0
dks3d11	1	0	0	0	0	0	0	7.70	0
dks3d12	1	0	0	0	0	0	0	0	7.70
dks4d[1-7]	7	0	0	0	.625	0	.4	0	0

Table 3.2: Priced Configuration Data Distribution

Disk	Total	UNIX	ORACLE	Branch	Account	Account	History	Log	Log
Name	# of	+swap	System&	&Teller	data	Index	Data	Data	Data
		Drives		Control files	data				(Mirror)
					% of data/disk				
dks4d8	1	0	0	0	0	0	.485	0	0
dks4d9	1	0	0	0	0	0	0	7.70	0
dks4d10	1	0	0	0	0	0	0	0	7.70
dks4d11	1	0	0	0	0	0	0	7.70	0
dks4d12	1	0	0	0	0	0	0	0	7.70
dks5d[1-7]	7	0	0	0	.625	0	.4	0	0
dks5d[8-9]	2	0	0	0	0	0	.485	0	0
dks5d1[0-2]	3	0	0	0	0	0	.485	0	0
dks6d[1-7]	7	0	0	0	.625	0	.4	0	0
dks6d8	1	0	0	0	0	0	.485	0	0
dks6d9	1	0	0	0	0	0	0	7.70	0
dks6d10	1	0	0	0	0	0	0	0	7.70
dks6d11	1	0	0	0	0	0	0	7.70	0
dks6d12	1	0	0	0	0	0	0	0	7.70
dks70d[1-8]	8	0	0	0	.625	0	.4	0	0
dks70d9	1	0	0	0	0	0	.485	0	0
dks70d1[0-2]	3	0	0	0	0	0	.485	0	0
dks71d[1-7]	7	0	0	0	.625	0	.4	0	0
dks71d8	1	0	0	0	0	0	0	7.70	0
dks71d9	1	0	0	0	0	0	0	0	7.70
dks71d10	1	0	0	0	0	0	0	7.70	0
dks71d11	1	0	0	0	0	0	0	0	7.70
dks71d12	1	0	0	0	0	0	0	7.70	0
dks71d13	1	0	0	0	0	0	0	0	7.70
dks71d1[4-5]	2	0	0	0	0	0	.485	0	0
dks72d[1-7]	7	0	0	0	.625	0	.4	0	0
dks72d8	1	0	0	0	0	0	0	0	7.70
dks72d9	1	0	0	0	0	0	.485	0	0
dks72d1[0-2]	3	0	0	0	0	0	.485	0	0
dks73d[1-7]	7	0	0	0	.625	0	.4	0	0
dks73d8	1	0	0	100.00	0	0	0	0	0
dks73d9	1	0	0	0	0	0	.485	0	0
dks73d1[0-2]	3	0	0	0	0	0	.485	0	0
dks74d[1-8]	8	0	0	0	.625	0	.4	0	0
dks74d9	1	0	0	0	0	0	.485	0	0
dks74d1[0-2]	3	0	0	0	0	0	.485	0	0
dks75d[1-8]	8	0	0	0	.625	0	.4	0	0
dks76d[1-8]	8	0	0	0	.625	0	.4	0	0
dks77d[1-8]	8	0	0	0	.625	0	.4	0	0
dks7d[1-7]	7	0	0	0	.625	0	.4	0	0
dks7d8	1	0	0	0	0	0	0	7.70	0

Table 3.2 Priced Configuration Data Distribution (continued)

Clause 4 Scaling Rules

4.1 Clause 4 Related Items

There are no Clause 4 Related Items required by the Full Disclosure specification. However, Clause 4 specifies scaling rules and that information is provided here as the appropriate place to describe the database size and scaling information.

4.1.1 Database Scaling, and Row Occurrences

The database was populated with the required number of row occurrences for the Account, Branch, and Teller tables to measure for 2200 tpsA. These numbers are listed in Table 4.1

The specific code used to create and populate these tables may be found in Appendix B. Details of the space calculated for the History table and log files may be found in Appendix D.

Table 4.1: CHALLENGE XL Server and ORACLE7 Required Row Occurrences

Table	Occurrences
Branch	2,200
Teller	22,000
Account	220,000,000

Clause 5 Distribution, Partitioning, and Transaction Generation

5.1 Random Number Generator

The method of verification of the random number generator should be described.

The UNIX function lrand48 was used to generate a pseudo-random number used as account identifiers and delta amounts in each TPC Benchmark™ A transaction.

This routine generates pseudo-random numbers using a well-known linear congruential algorithm. The algorithm will generate unique sequences of numbers if provided with a unique seed for each sequence. The RTE constructed a seed for each simulated teller using this formula:

```
gettimeofday (&tv, (struct timezone *)0);
srand48 (teller * tv.tv_usec);
srand (teller * tv. tv_usec);
```

At the end of each TPC Benchmark™ A run, the seeds for all tellers were checked and verified to be unique.

In addition, the History and success files were randomly searched by the auditors for duplicates and/or patterns that would indicate the random number generator had effected any kind of discernible pattern. None were found.

5.2 Horizontal Partitioning

Vendors must clearly disclose if horizontal partitioning is used. Specifically, vendors must satisfy the following:

1. *Describe textually the extent of transparency of the implementation.*

2. *Describe which tables / files were accessed using partitioning.*

3. *Describe how partitioned tables / files were accessed.*

The account and History files were horizontally partitioned. The partitioning was completely transparent to the application. The DBMS completely controlled the access to all portions of the table files regardless of where they were stored on disk. The complete description of the physical positioning of the tables may be found in Clause 3 under **Database Design,** and **Distribution and Partitioning.**

The sponsor must disclose the percentage of remote and home transactions, percentage of remote and foreign transactions, if applicable, and the actual distribution of accounts across the nodes, if applicable.

The percentage of remote and home transactions during the measured benchmark runs were 15% and 85% respectively. The benchmark was run on a single system.

Clause 6 Response Time

6.1 Benchmark Performance

Report all the data specified in Clause 6.6, including measured and reported tpsA, maximum and average response time, as well as performance curves for number of transactions vs. response time (see clause 6.6.1) and response time distribution (see clause 6.6.2). Also, the sponsor must include the percentage of home and remote transactions, the number and percentage of in-process transactions, and the percentage of remote and foreign transactions, if applicable.

Table 6.1 contains the statistics required by the above clause.

Table 6.1: CHALLENGE XL Server and ORACLE7 Performance Statistics

Measured tpsA	2049.71	tps-A
Reported tpsA	2049.71	tps-A
90th percentile Response time	1.40	seconds
Maximum Response Time	21.260	seconds
Average Response Time	1.169	seconds
Percent of Home transactions	85.00	%
Percent of Remote transactions	15.00	%
Measured completed transactions	4509372	
Number of in flight transactions	2344	
Percentage of in flight transactions	.052	%

6.1.1 Response Time

The distribution of response times for the transactions in the benchmark test are shown below in Figure 6.1.

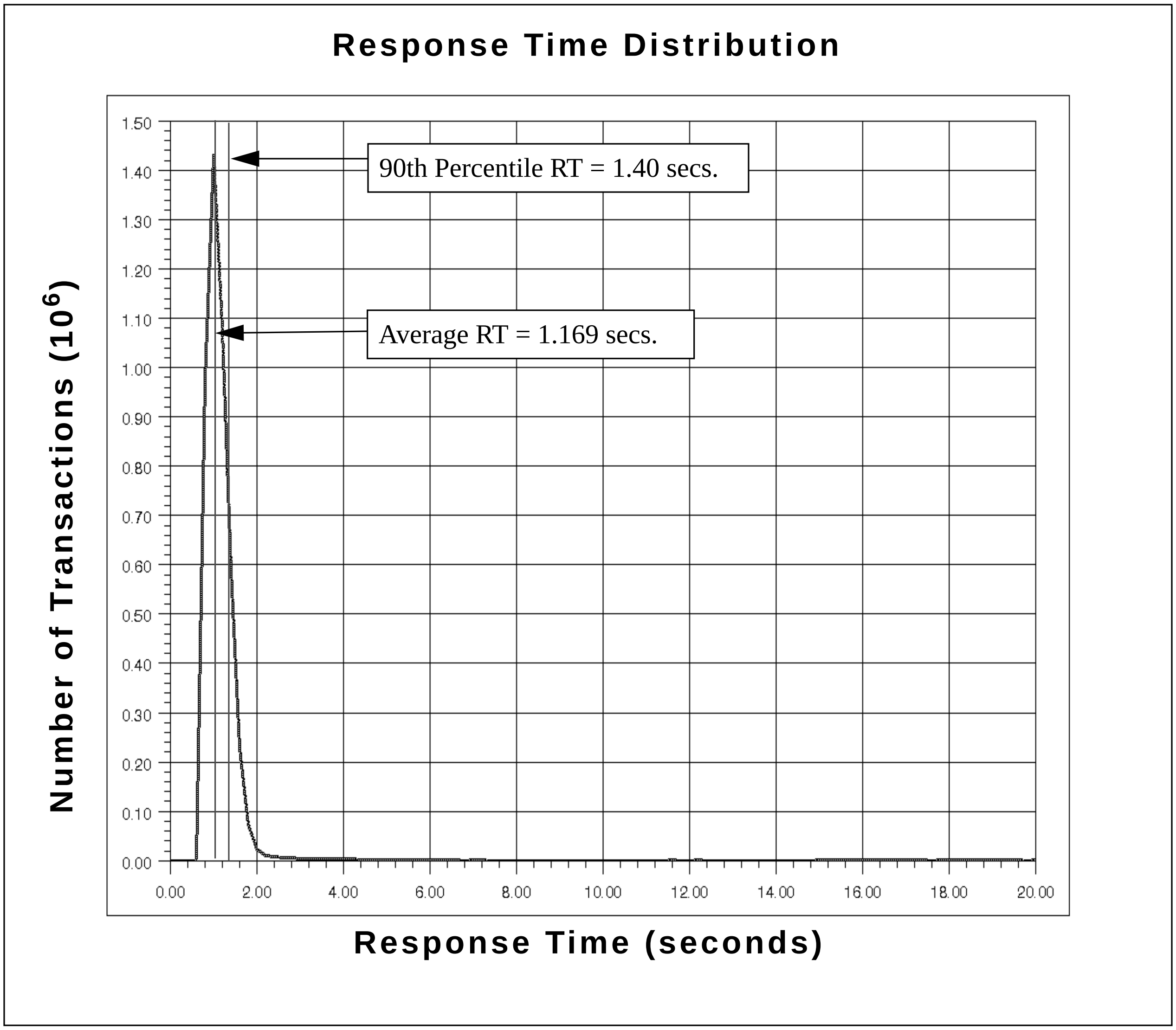


Figure 6.1: CHALLENGE XL Server and ORACLE7 Response Times

6.1.2 Throughput (tpsA) vs. Response Time

The throughput (tpsA) vs. 90th percentile response time graphs are shown below in Figure 6.2. The graph shows the average response time at 100%, 80%, and 50% of the reported tpsA throughput. The 80% and 50% throughput rates were obtained by varying the think time. Everything else in the 80% and 50% tests was identical to the 100% test.

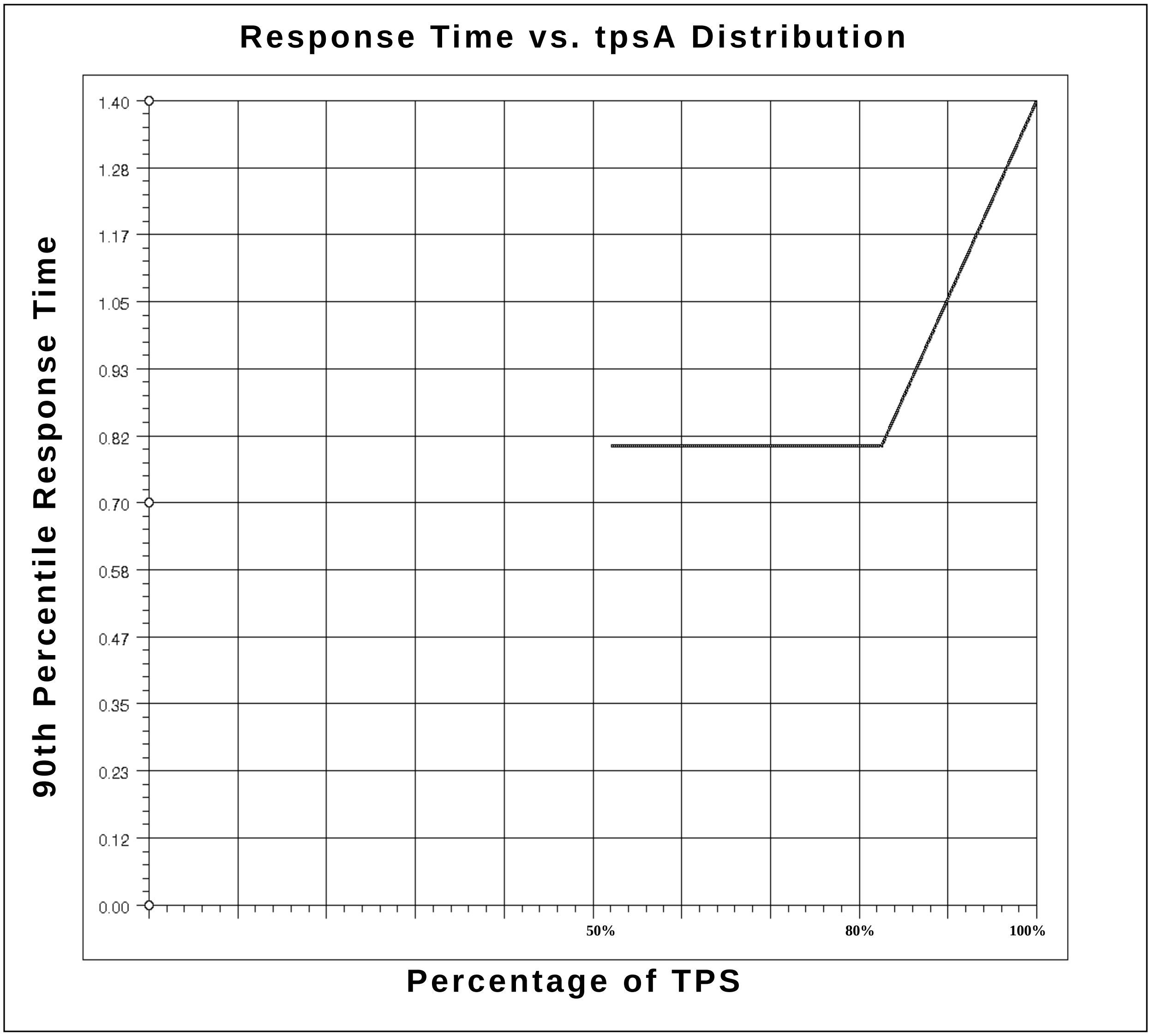


Figure 6.2: CHALLENGE XL Server and ORACLE7 Response Time vs. tpsA

Clause 7 Duration of Test

7.1 Steady State

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval should be described.

The transaction throughput rate (tpsA) was measured during trial runs to determine the average time required to start all processes and begin a sustained rate of throughput. This ramp up interval was also verified by performance monitoring information. The ramp up interval of twenty (20) minutes was sent as a parameter to the benchmark application to assure that the measured interval was started after a steady state was established.

7.2 Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log records, etc. as required by Clause 7.2), actually occurred during the measurement interval.

During the measurement interval, the ORACLE7 RDBMS reads one account block into the buffer cache for every transaction. On average, one modified account block was written from the shared buffer cache for every transaction, but this write was only necessary to free space in the shared buffer cache, not to commit the transaction. Modified database buffers migrated to disk on a “least recently used” basis independent of transaction commits. In addition, every block modification was protected by redo log records. These redo log records were written to the redo log buffer (in memory), which were flushed to a redo log file on disk either when the transaction committed or when the redo log buffer became full.

During a checkpoint, all modified blocks in the shared buffer cache which had not been written to disk since the last checkpoint were physically written to disk. A single checkpoint was performed during the measurement interval.

The performance of the TPC Benchmark A transaction was improved by using the BEGIN_DISCRETE_TRANSACTION procedure (See Appendix A). This procedure streamlines transaction processing so that short, non-distributed transactions can execute more rapidly.

During a discrete transaction, all changes made to any data were deferred until the transaction committed. Redo information was generated, but was stored in a separate location in memory. When the transaction issued a commit request, the redo information was written to the redo log file (along with other group commits) and the changes to the database block were applied directly to the block. Once the commit completed, control was then returned to the application.

Notice the loop construct in the transaction profile included in Appendix A. The TPC-A transaction was implemented as a discrete transaction by calling the BEGIN_DISCRETE_TRANSACTION procedure before the first statement. Any error encountered during the processing of discrete transactions caused the pre-defined exception DISCRETE_TRANSACTION_FAILED to be raised. If this exception occurred, the TPC-A transaction was rolled back and re-executed as a normal transaction.

The discrete transaction is a fully documented performance feature in the ORACLE7 DBAGuide.

Software on the RTE machines emulates tellers typing account transaction information on a terminal and receiving replies. The input data is passed to a Tuxedo client program on one of the Client machines. There is one copy of the RTE program and one copy of the Tuxedo client program running for each simulated teller, connected via Unix TCP/IP sockets in a one-to-one correspondence.

Each client machine runs 10 copies of a Tuxedo server program. The Tuxedo servers call the Oracle OCI library function to communicate with the RDBMS via Oracle SQL*NET software. The client programs add transactions to message queues for the server programs to pick up and pass to the RDBMS. When a server receives a reply from the RDBMS it passes the reply back to the originating client, which in turn passes it back to the RTE.

7.3 Reproducibility

A description of the method used to determine the reproducibility of the measurement results.

The benchmark was executed multiple times and the reported throughput (tpsA) and residence time varied less than one point one (1.10) percent between the measured runs.

7.4 Measurement Period Duration

A statement of the duration of the measurement period for the reported tpsA (it should be at least 15 minutes and no longer than 1 hour).

Each measured run was executed for a total of 78 minutes. This included 20 minutes of ramp up time and 36 minutes of steady state. The measurement interval was 37 minutes and included 1 checkpoint. The checkpoints were set to begin every 39 minutes.

The graph demonstrating steady state is shown below.

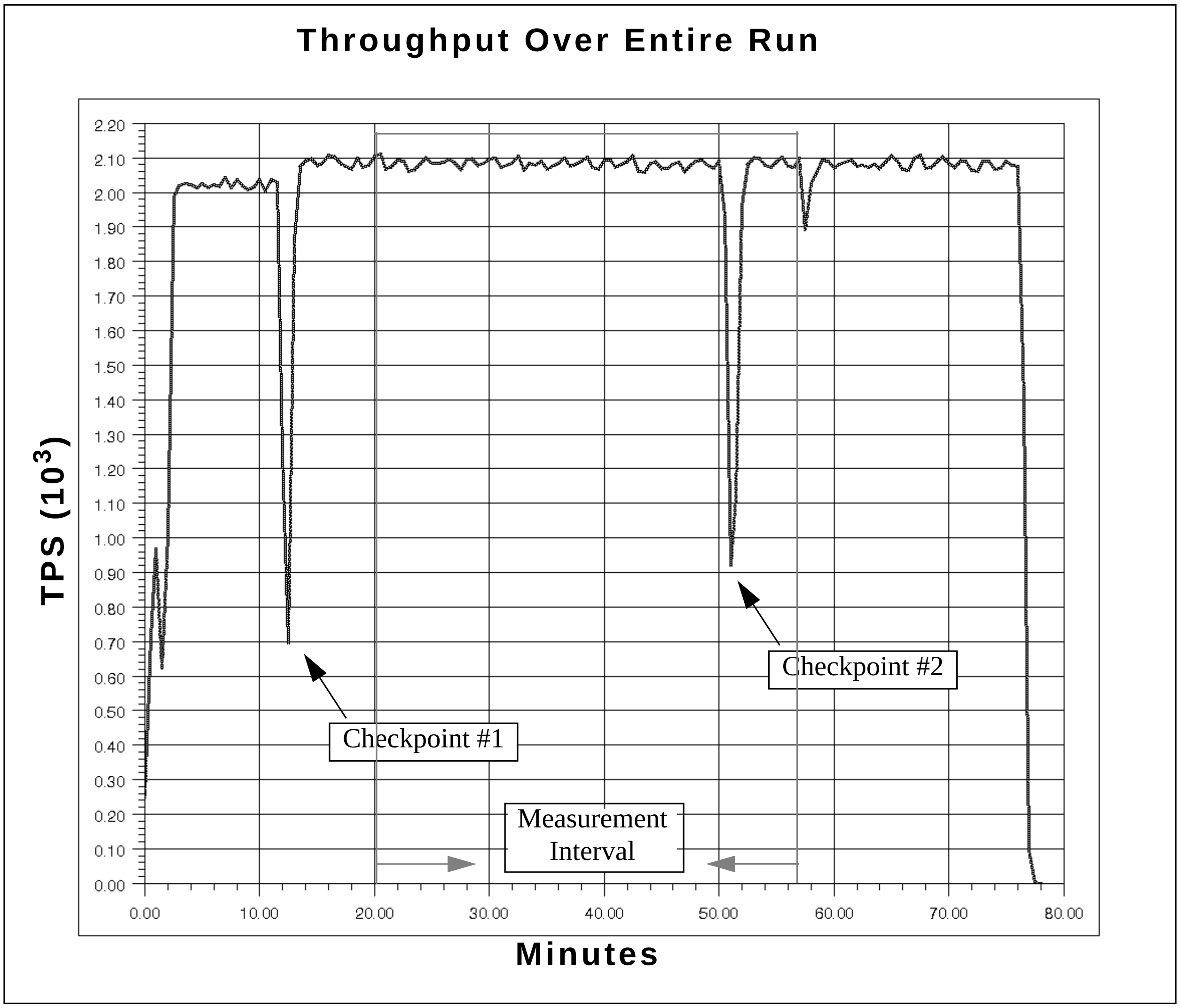


Figure 7.1: CHALLENGE XL Server with ORACLE7 Throughput tpsA

Clause 8 SUT & Driver

8.1 RTE

1. The name of the RTE and whether it is commercially available or proprietary

The RTE used is proprietary. It was jointly developed by Oracle Corporation and Silicon Graphics, Inc. Inputs to the RTE include the number of clients to initiate, how long they are to run, sleep time, TPS scaling, times for ramp up and ramp down, the type of transaction workload to run, tty delay, and a number of other parameters. Appendix A contains the entirety of the transaction portion of the driver software that implements the TPC-A transaction. Appendix F contains the Open Call Interface code for the section of the RTE used to control transaction submission and timing during ramp up, steady state, and ramp down.

2. The hardware on which the RTE runs.

The RTE runs on any MIPS ABI compliant platform. The machines that were actually used during the test were 20 Silicon Graphics Indy workstations.

3. The component(s) emulated by the RTE.

The RTE emulates all the terminals and terminal servers.

4. Commands to start the RTE including pertinent parameters.

The parameters to the RTE include the number of clients to initiate, and how long they are to run. (See item 10 below for a description). Each RTE machine started 1042 rte processes. Here is a sample command line to start one rte process.

rte jade 220 tpca conf 3489 2200 1042 8337 1 1 9.55 2200 1200 1200 2084 .58

5. The type of communication protocol used or simulated between the RTE and SUT.

The RTE communicates with the front-end clients of the SUT. Via telnet sessions, the RTE simulates terminal input, serial line delays to the terminal, ethernet delays, etc. Telnet protocol is implemented over TCP/IP using UNIX sockets.

6. The timing delays associated with the simulation of the components and the communication protocol used.

(See Section 8.2 Driver Functionality and Performance).

7. Generation of the success file (used for testing durability).

Whenever a transaction completes, the results of the transaction are written to a success file by the client process. This occurs during durability testing. After the run these files are processed to display the required information, such as account, branch, teller, amount, and amount delta.

8. The number of processes per simulated terminal.

There is one rte process per simulated terminal.

9. Generation of random numbers to show that no two simulated terminals will use the same pseudo-random sequence.

Every emulated terminal is assigned a unique teller. The branch to which this teller belongs is the branch row that is used 85% of the time and a random branch is picked 15% of the time using the function ‘lrand48’.

The random number generator is initialized with the teller # modified by the current time-of-day to ensure that no two users will generate the same random number sequence. Here is the code fragment that seeds the generators:

```
gettimeofday (&tv, (struct timezone *)0);

srand 48 (teller * tv.tv_usec);

srand (teller * tv. tv_usec);
```

The account number is then computed as a random account at that particular branch using the ‘lrand48’ function.

10. Listing of input scripts and parameter files to the RTE.

Input to the RTE is through command-line arguments. Here is a complete list of the arguments. The arguments from the sample line in Item 4 are shown in parentheses:

host	Client Hostname	(jade)
port	Port Number	(220)
name	Login Name	(tpca)
config	Configuration ID	(conf)
runname	Run Identifier	(3489)
timelimit	Run Time	(2200)
nproc	Number of Users	(1042)
proc_no	Current Process Number	(8337)
ncpu	Number of CPUs on Client	(1)
nsvr	Number of Servers	(1)
thinktime	Think Time	(9.55secs)
mult	Database Scale	(2200)
ramp_up	Ramp Up Interval	(1200secs)
ramp_down	Ramp Down Interval	(1200secs)
starttime	Start Time of Run	(2084)
termdelay	Emulated Terminal Delay	(0.58secs)

The rte processes were started by a script on each RTE machine that varied two parameters. Port numbers cycled from 220 to 249. proc_no increased by 1 for each process created. One parameter to that script was the first proc_no to use, to ensure proc_no remained unique across all rte processes on all RTE machines.

11. Algorithm used to generate transaction input and a sample of that input.

The RTE program sends strings like this one to the client program on a client machine (all one line):

```
0013,00123,001294403,0000679623,xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxXXXXXXXXXX
```

The client program receives the string, packages it as a transaction, and sends the transaction to a transaction monitor, which forwards it to the RDMBS.

The code used to generate the transactions is shown in Appendix F.

12. Algorithm used to determine delay times between transactions.

The code used to determine delay times is shown in Appendix F.

13. Benchmark sequencing including ramp-up period, steady state measurement window(s), and transaction success/failure determination and recording.

The code used to determine benchmark sequencing is shown in Appendix F.

14. A list and brief description of the data that are collected and the reduction process of that data to determine the results.

Each client process collects:

- The count of transactions performed (tr_count),
- The count of the transactions that complete in less than 2 seconds (tr_fast),
- The count of the transactions that start during the measurement period but do not complete (in_flight),
- The sum of the response times (tr_sum),
- Think times (tk_sum),
- Elapsed times,
- Minimum and Maximum values of the think times,
- Minimum and Maximum response times.

This data is collected during steady state operation.

The RTE uses this data to compute the cumulative data for the run:

```
tps = tr_count / timelimit;
fast_percent = 100* tr_fast / tr_count;/* % completed in 2 secs */
avg_resp = tr_sum / tr_count;/* average response time */
avg_think = tk_sum / tr_count;/* average think time */
```

The histogram of response times and think times are obtained from the corresponding arrays timing_buckets and think_buckets shown in Appendix F.

The number of transactions performed in each 15 second interval is obtained by grouping the history records according to the time stamp and counting the records that fall into each interval.

8.2 Driver Functionality and Performance

A proof that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The sponsor must list all hardware and software functionality of the driver and its interface to the SUT.

In the priced configuration the appropriate number of terminals are connected equally across the 20 Indy workstations that are used as client systems. This connection is performed with 652 Specialix MTS and 1956 MTA terminal servers. Transactions are submitted from the terminals (via terminal servers), to the client machines which are front end clients of the Database server. The terminal servers are evenly distributed across 4 Ethernets, each of which is connected to 1/4 (i.e., 5) of the client machines. Each client has two ethernet ports; the second port connects to one of 4 Ethernets connected to the DBMS server machine. The DBMS front end software running on the clients connect to the DBMS servers via network sockets.

Clause 8.6.4.1 of the TPC Benchmark™ A specification allows a test sponsor to emulate the terminal network if the proposed solution makes it uneconomical to perform the work in question directly on such a terminal network.

Due to the exceptionally large number of terminals, terminal servers, terminal server expansion cards, and cables involved in such a large system, Silicon Graphics Computer Systems emulated the terminal network from terminals up to (but not including) client machines with the RTE machines.

The RTE drivers emulate the system of Wyse WY-30+ terminals connected via RS-232 cables to Specialix MTS terminal servers (including MTA expansion cards), which open TCP/IP telnet sessions to client Indy workstations over Ethernet.

An experiment was performed to determine how much additional delay in response time is introduced by using an actual terminal server. Two Specialix MTS terminal servers, each with three MTA expansion boards, were connected together. The 32 serial lines of one server were connected back-to-back with the 32 lines of the other. Both were configured for 38.4 Kbaud and connected to one of the driver-client LANs. The RTE on one driver machine was configured to go through this setup for its first 32 emulated terminals.

The average response time for terminals going through the terminal servers was calculated separately from the response time for terminals going through the normal benchmark setup under the load of a full TPC-A benchmark run. The maximum observed difference was 0.576 seconds.

	Specialix	Emulated	Combined
Users	1	1	2
TPS	0.15	0.15	0.3
Response Time	0.475	0.027	0.251
	difference in response time: 0.452 sec		
	average cycle time: 6.667 sec		
Users	16	1184	1200
TPS	2.94	120.67	123.61
Response Time	0.856	0.672	0.676
	difference in response time: 0.184 sec		
	average cycle time: 9.684 sec		
Users	32	1168	1200
TPS	2.79	121.74	124.53
Response Time	1.144	0.568	0.581
	difference in response time: 0.576 sec		
	average cycle time: 9.612 sec		
Users	32	20,808	20,840
TPS	3.00	2057.15	2060.15
Response Time	1.608	1.215	1.215
	difference in response time: 0.383 sec		
	average cycle time: 10.116 sec		

Figure 8.1: Measured timings for Specialix concentrators and RTE program

By including a minimum 0.58 second delay in the rte program, the difference in performance between the emulated and priced systems is accounted for.

8.3 Network Bandwidth

If the SUT contains a WAN or a LAN network, its bandwidth should be specified.

The Twisted Pair and Thinnet Ethernets used in the LAN comply with the IEEE 802.3 standard and have a bandwidth of 10 Mbps.

8.4 Think Time

The sponsor must disclose the mean and maximum think times and a graph of the distribution of think times.

The mean and maximum think times are 9.569 seconds and 99.294 seconds respectively. The think time distribution is graphed in Figure 8 below.

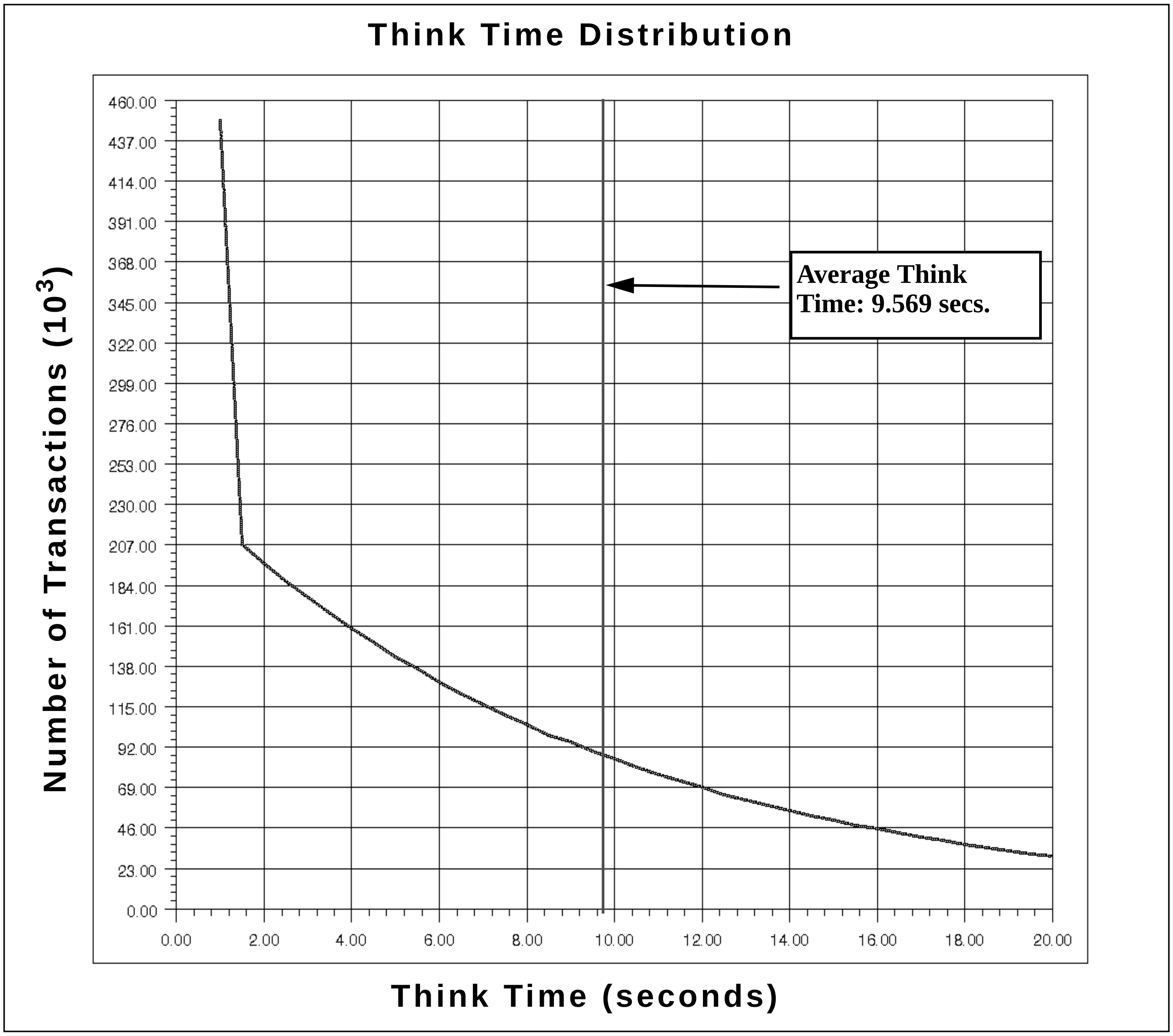


Figure 8.2: CHALLENGE XL Server with ORACLE7 Think Time Distribution

Clause 9 Pricing

9.1 System Pricing

A detailed list of hardware and software used in the priced system. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery data. If package-pricing is used, contents of the package must be disclosed.

The total price of the entire configuration is required including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used shall be disclosed.

See Priced Configuration Table 9.1

9.1.1 CHALLENGE XL Server

The CHALLENGE XL Server system consists of:

- 32 R4400SC CPUs at 150 MHz each with 4MB of combined secondary cache
- 16K data and 16K instruction primary cache
- 1 GB of main memory with 8-way interleaving
- 3 POWERchannel-2 I/O boards each with an Ethernet interface with 6 additional SCSI-2 cards for a total of 24 FAST/-WIDE SCSI-2 channels

- 265 disk drives of 1.92 GB formatted capacity each
- 2GB DAT tape drive, QIC-150 cartridge tape, and CD-ROM
- 2 EFAST Ethernet boards

9.1.2 Support Pricing

The five year support pricing for CHALLENGE XL Server

$539,275.00

The five year support pricing for ORACLE7

$501,312.00

9.1.3 Priced System Configuration

The hardware, software, and support/maintenance products priced in this benchmark, are detailed below in Table 9.1. Also included in the table are the measured tpsA and calculated price/tpsA.

9.1.4 Discounts

The following generally available discounts to any buyers with like conditions were applied to the priced configuration:

- a 16% Silicon Graphics Computer Systems Volume End User Discount was applied to the CHALLENGE XL Server configuration.
- a 5% Silicon Graphics Computer Systems Volume End User Discount was applied to the Indy Client configuration.
- an 18% ORACLE Corporation Volume End User Discount was applied to the Oracle license fees.
- a 10% ORACLE Corporation Volume End User Discount was applied to the Oracle maintenance fees.

9.2 Availability

T*he delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available*

Products used in the benchmark are currently available except for

The CHALLENGE XL Server used in the benchmark and identified by Order Number R-45832-S4 is orderable now and will be deliverable on December 30, 1994.

The IRIX (UNIX SVR4) operating system which was used on the SUT, is a pre-released version of IRIX which will be released on December 30, 1994.

ORACLE7 version 7.0.15.4.2 will be available May 30, 1994.

9.3 Measured tpsA

A statement of the measured tpsA and the calculated price/tpsA.

See Priced Configuration Table 9.1

9.4 Priced Storage Requirements

The basis for the calculation to determine the additional storage space required in Clause 9.2.3.1 must be included

The hardware necessary to meet the storage requirements of Clause 9.2.4 was calculated based upon the number of history records stored per page and the measured transaction rate. The log file storage required was calculated, based on statistics which are generated by ORACLE7 for each run. Details of the files that these calculations are based on are included in Appendix D.

Order Number	Description	Quantity	Unit Price	Extended Price	Support (5 years)
CHALLENGE XL Server					
R-45832-S4	32-CPU CHALLENGE XL Server, 4MB cache	1	$876,700	$876,700	$277,750
FTO-64UP512	First 512 MB High Dens Mem, 1 IMB	1	63,520	63,520	8,525
H4-512-4-ADD	Addt'l 512MB high Dens. Mem, 2 IMBs	1	82,800	82,800	13,975
P-S-B224	CHALLENGEvault XL 224GB Disk Bundle	2	560,000	1,120,000	184,800
P-S-B64	CHALLENGEvault XL 64GB Disk Bundle	1	196,250	196,250	26,400
P8-S-2	2GB SCSI-2 FAST/WIDE Disk	10	6,900	69,000	8,250
HU-PC2	POWERChannel-2 I/O Controller	2	12,000	24,000	8,500
P-S-HIO	SCSI-2 FAST/WIDE Interface Card	6	2,500	15,000	5,400
P8-QIC-CD	150 MB QIC tape & CD-ROM	1	2,000	2,000	1,700
P8-DAT	2GB DAT internal drive	1	2,500	2,500	1,125
P-TER2	110 VAC Programming Terminal	1	1,500	1,500	600
DK-C2-001	Destination Kit for XL Series	1	0.00	0.00	0.00
CC4-EFAST-2.0.1	Addt'l Ethernet Interface for CHALLENGE	2	5,700	11,400	2,250
DK-T2-001	Destination Kit for CHALLENGEvault XL	5	0.00	0.00	0.00
SC4-S4D-5.3	Operating System Software and Manuals	1	0.00	0.00	0.00
SC4-IDO-5.3	IRIX development options for IRIX 5.3	1	1,200	1,200	0.00
CS-SWCARE-DEV	Software options support (incl. IDO)	1	0.00	0.00	0.00
		TOTAL Server		**2,465,870**	**539,275**
Indy					
CH-S100	Indy, 100MHz, R4000SC, 1GB system disk	20	13,495	269,900	149,000
HU-M128A	128MB memory upgrade for Indy	40	18,000	720,000	0.00
CC2-E++-1.0	GIO Bus Ethernet Card	20	625	12,500	0.00
	32MB return to factory for Indy	20	-3,000	-60,000	0.00
		TOTAL Client		**942,400**	**149,000**
Communications & Terminals					
	†Specialix MTS	718	678	486,804	17,950
	†Specialix MTA 8-port expanders	2152	228	490,656	53,800
	Wyse WY-30+ terminals	20,840	189	3,938,760	729,400
	‡Anixter 8-port 10BaseT Hub/BNC	6	375	2,250	0.00
	‡Anixter Ethernet/IEEE Transceivers BNCTap	22	49	1,078	0.00
		TOTAL Comms.		**4,919,548**	**801,150**
ORACLE Software					
	ORACLE7 (448 named users)	1	358,400	358,400	215,040
	Procedural Option (448 named users)	1	71,680	71,680	43,008
	SQL*Net (448 named users)	1	71,680	71,680	43,008
	TCP/IP protocol driver (448 named users)	1	53,760	53,760	32,256
	SQL*Net (64 named users)	20	8,000	160,000	96,000
	TCP/IP protocol driver (64 named users)	20	6,000	120,000	72,000
Tuxedo Software					
	Tuxedo 4.2.1 (>10,001 users)	20,840	20	416,800	208,400
	Development system	1	380	380	190
		TOTAL Software		**1,252,700**	**709,902**
Discounts					
	Oracle Volume Discount			-150,393.60	-50,131.20
	Silicon Graphics CHALLENGE XL Discount			-394,539.20	0.00
	Silicon Graphics Indy Discount			-47,120.00	0.00
		TOTAL Discounts		**-592,052.80**	**-50,131.20**
Total Hardware and Software Costs					**11,137,661**
tps-A					**2049.71**
$/tps-A					**$5,433.77**

†includes 10% spares
‡includes 10% spares and 5 year maintenance

Table 9.1 Priced Configuration

Table 9.2: Bundled Item Descriptions

Order Number	Quantity	Description
R-45832-S4	1	32-CPU CHALLENGE XL Server
	1	CHALLENGEXL rack chassis
	32	150MHz MIPS R4400SC CPUs
	1	4 MB SRAM Secondary Cache per CPU
	1	2 GB SCSI-2 FAST/WIDE Differential System Disk
	1	64MB memory board
	1	POWERchannel-2 I/O board
	2	SCSI-2 channels
	2	SCSIBOX-2 disk tray
	1	ethernet channel
	1	parallel port
	3	RS-232C ports
	1	RS-422 port
P-S-B224	2	CHALLENGEvault XL 224GB disk bundle
includes	2	CHALLENGEvault rack
	14	SCSIBOX-2, 8 disk enclosure
	112	1.92GB disk drives
P-S-B64	1	CHALLENGEvault XL 64GB disk bundle
includes	1	CHALLENGEvault rack
	4	SCSIBOX-2, 8 disk enclosure
	32	1.92GB disk drives

Clause 10 Full Disclosure Checklist

10.1 General Items

A statement verifying the sponsor of the benchmark and any other companies who have participated.

The benchmark is being sponsored by Silicon Graphics Computer Systems, the hardware vendor, and ORACLE Corporation, the supplier of the database management system used.

Program listing of application code and definition language statements for file/tables.

Appendix A contains a listing of the application programs which were written in the “C” language. Appendix B contains the Bourne shell scripts, “C” source code and SQL scripts which were used to create and load the benchmark database.

Settings for all customer-tunable parameters and options which have been changed from defaults found in actual products; including but not limited to: Database options; Recovery/Commit options; Consistency/-Locking options; System parameters; application parameters, and configuration parameters. Test sponsors may optionally provide a full list of all parameters and options.

A listing of all modified operating system parameters and all database parameters configured during the benchmark is given in Appendix C.

Configuration diagrams of both the benchmark configuration and the priced system, and a description of the differences.

A diagram of the SUT and the priced configuration are in Clause 3.

10.2 Clause 2 Related Items

Results of the ACIDity tests must describe how the requirements were met. If a database different from that which is measured is used for durability tests, the sponsor must include a statement that durability works on the fully loaded and fully scaled database.

The ACIDity tests performed are described in Clause 2.

10.3 Clause 3 Related Items

The distribution across storage media of ABTH files/tables and all logs must be explicitly depicted.

Provide two functional diagrams which show CPUs, storage devices, communications lines, terminals, and the interconnections between these components. The first diagram must correspond to the benchmark configuration and the second diagram must correspond to the 90-day priced configuration. A separate pair of diagrams must be provided for each reported result.

As part of each diagram, show the percentage of the total physical database which resides on each storage device for each of the ABTH files and logs. For the benchmark configuration, show database allocation during 8-hour steady state. For the 90-day priced configuration, show database allocation including storage of 90 days of history records. Data which are duplicated (e.g., mirrored) on more than one device must be clearly labeled to show what is duplicated and on which devices.

The distribution of the ABTH files/tables, log, and system files is depicted in Tables 3.1 and 3.2 in Clause 3. A diagram of the SUT and the priced configuration are also included in Clause 3.

A description of how the database was populated, along with sample contents of each ABTH file/table to meet the requirements described in Clause 3.

Clause 3 contains the details of the Logical Database Design. Samples of the ABTH file contents are shown in Appendix B.

A statement of the type of database utilized.

The benchmark was conducted using ORACLE7, a standard relational database management system which is a product of ORACLE Corporation.

10.4 Clause 5 Related Items

The method of verification of the random number generator should be described.

The random number generator used is described in Clause 5.

Vendors must clearly disclose if horizontal partitioning is used. Specifically, vendors must: describe textually the extent of transparency of the implementation; describe which tables/files were accessed using partitioning; and describe how partitioned tables/files were accessed.

The intent of this clause is that details of non-transparent partitioning be disclosed in a manner understandable to non-programmer individuals (through use of flow charts, pseudo code, etc.).

Disk partitioning is described in Clause 3.1.1. as well as tables 3.1 and 3.2.

The sponsor must disclose percentage of remote and home transactions, percentage of remote and foreign transactions, if applicable, and the actual distribution of accounts across the nodes, if applicable.

The account information is given in Clause 6, Table 6.1

10.5 Clause 6 Related Items

Report all the data specified in Clause 6.6, including reported tpsA, maximum and average response time, as well as performance curves for tpsA versus response time and response time distribution.

Response data, including the required graphs, are described in Clause 6. The graph of total system throughput is given in Clause 7.

10.6 Clause 7 Related Items

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval should be described.

That the SUT had achieved steady state was determined by observing the transaction processing rate at 30 second intervals.

A description of how the work normally performed during a sustained test actually occurred during the measurement interval.

The description of all work performed, including checkpoints, is detailed in Clause 7.

A description of the method used to determine the reproducibility of the measurement results.

The benchmark result was reproduced with a variance of less than 1.10%.

A statement of the duration of the measurement period for the reported tpsA.

The measurement period was 36 minutes.

10.7 Clause 8 Related Items

Disclose the following information related to the RTE: name of RTE and whether it is commercially available or proprietary, hardware on which the RTE runs, components emulated by the RTE, commands to start the RTE including pertinent parameters, type of communication protocol used or simulated between the RTE and SUT, timing delays associated with the simulation of the components and the communication protocol used, generation of the success file, number of processes per simulated terminal (one process for each terminal or one process per multiple terminals), generation of random numbers to show that no two simulated terminals will use the same pseudo-random sequence, listing of inputs scripts and parameter file to the RTE, algorithm used to generate transaction input and a sample of that input, algorithm used to determine delay times between transactions, benchmark sequencing including ramp-up period, steady state measurement window(s), and transaction success/-failure determination and recording, a list and brief description of the data that are collected and the reduction process of that data to determine the results.

The driver used in the benchmark is proprietary to ORACLE Corporation. It resided on the driver/RTE machines and is described in Clause 8 and in Appendix F. Additional information regarding the above list is also described in Clause 8.1

A proof that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The sponsor must list all hardware and software functionality of the driver and its interface to the SUT.

The driver information is described in Clause 8.2.

If the SUT contains a WAN or a LAN network, its bandwidth should be specified. The sponsor must describe the network configuration per clause 8.6.5.

Clause 8.2 gives the network bandwidth. Clause 8.3 describes the network configuration for the SUT.

The sponsor must disclose the mean and maximum think times and a graph of distribution of the think times.

Think time information is disclosed in Clause 8.4 and depicted in Figure 8.2.

10.8 Clause 9 Related Items

A detailed list of hardware and software used in the priced system must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed.

Priced system information is described in Clause 9 and in Appendix G.

The total price of the entire configuration is required including: hardware, software, and maintenance changes. Separate component pricing is recommended. The basis of all discounts shall be disclosed.

All pricing information is contained in Clause 9

The delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

Availability information is provided in Clause 9.3.

A statement of the measured tpsA, and the calculated price/tpsA.

The CHALLENGE XL Server was measured at 2049.71 tpsA at a price of $5,433.77/tpsA.

The basis for the calculation to determine the additional storage space required in Clause 9.2.3.1 must be included.

Appendix D contains storage space calculations.

10.9 Clause 11 Related Items

If the benchmark has been independently audited, then the auditor's name, address, phone number, and a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included, specifying when the complete audit report will become available and whom to contact in order to obtain a copy.

Clause 11 contains auditor information.

10.10 FDR Availability

The full disclosure report is to be readily available to the public at a reasonable charge, similar to charges for similar documents by that test sponsor. The report is to be made available when results are made public.

For copies of this report contact:

SGI Express
315 North State Street
Orem, UT 84057
ph: 1-800-336-0264
title: TPC Benchmark™ A Full Disclosure Report

Clause 11 Related Items

11.1 Independent Auditing

If the benchmark has been independently audited, then the auditor's name, address, phone number and a brief audit summary report indicating compliance must be included in the full disclosure report.

The Silicon Graphics Computer Systems' CHALLENGE XL Server and ORACLE7 benchmark was independently audited by Performance Metrics, Inc. The attestation letter is included in Appendix E.

Appendix A: Application Source Code

Silicon Graphics Computer Systems' implementation of the TPC Benchmark™ A consists of C programs that provide both driver and transaction functions. The following listings are those C programs used for these functions.

Client Application

```
/*==================================================================+
 |         Copyright (c) 1992  Oracle Corp, Belmont, CA         |
 |                   All Rights Reserved                   |
 +==================================================================+
 | FILENAME
 |    t_cli.c
 | DESCRIPTION
 |    TPC-A Tuxedo client process
 +==================================================================*/

#include <stdio.h>
#include <termio.h>
#include <signal.h>
#include <sys/types.h>
#include "atmi.h"

struct xactinfo
{
   long teller_no;
   long branch_no;
   long account_no;
   long amount;
   double balance;
};

struct xactinfo *xact;
struct termio prevtty, newtty;
int cpu_no;
int timeoutcount = 0;
int srvnum;
char servicename[10];

quit()
{
    ioctl(0, TCSETA, &prevtty); /* Restore terminal parameters */

    exit(1);
}

void terminated(signo)
    int signo;
{
    if (signo != SIGHUP)
        userlog("Client terminated by signal %d\n", signo);

    if (xact != NULL)
        tpfree((char *)xact);  /* Free the Tuxedo message buffer. */

    tpterm(); /* Perform Tuxedo client exit. */

    ioctl(0, TCSETA, &prevtty); /* Restore terminal parameters */

    exit(0);
}

main(argc, argv)
    int argc;
    char * argv[];
{
    char inbuf[512], outbuf[512];
    char filler[165];
    int olen;
    int i;

    signal(SIGHUP,  terminated);
    signal(SIGINT,  terminated);
    signal(SIGQUIT, terminated);
    signal(SIGKILL, terminated);
    signal(SIGPIPE, terminated);
    signal(SIGTERM, terminated);

    timeoutcount = 0;

    for (i = 0; i < 160; i++)
        filler[i] = 'X';
    filler[160] = '\0';

    /* Get terminal parameters */
    ioctl(0, TCGETA, &prevtty);
```

```
/* Turn off echo */
newtty = prevtty;
newtty.c_lflag &= ~ECHO;
ioctl(0, TCSETA, &newtty);

/* Initialize Tuxedo */
if (tpinit(NULL) == -1)
{
    userlog("Client: tpcinit failed. Quitting.\n");
    quit();
}

/* Allocate Tuxedo message buffer.  */
if ((xact = (struct xactinfo *)
    tpalloc("CARRAY", NULL, sizeof(struct xactinfo))) == NULL)
{
    userlog("Client: tpalloc failed.\n");
    tpterm();      /* Perform Tuxedo client exit. */
    quit();
}

/* synchronize with RTE */

write(1, "Go TPC-A", 8);

/* Read message from rte. */
while (gets(inbuf) == NULL);

/* Extract cpu number and Tuxedo service number from the rte's message. */
sscanf(inbuf, "%d%d", &cpu_no, &srvnum);

/*
**  Use the service number to create Tuxedo service name for this client.
**  A given client will send all of its messages to a single server.
**  There is one server performing each service.
*/
sprintf(servicename, "TPCA%d", srvnum);

/*
**  Tell rte to start sending transactions.
*/
write(1, "Go TPC-A really", 15);

/*
** Loop, reading messages from rte and sending requests to the
** Tuxedo server.  Quit when the rte sends a negative branch id.
*/
while (1)
{
```

```
/*
**  Read message from rte.  The client must read at least 100 bytes
**  from the rte (Clause 8.4.2).  Here the client uses gets to read
**  up to a new-line character.  The rte is responsible for sending
**  messages of the correct size.
*/
while (gets(inbuf) == NULL);

/*
**  Extract the branch id, teller id, account id, and update amount
**  from the rte's message.  Store these values in the Tuxedo
**  message buffer.
*/
sscanf(inbuf, "%d,%d,%d,%d", &(xact->branch_no), &(xact->teller_no),
    &(xact->account_no), &(xact->amount));

/*
**  If the branch id is negative, exit.  This is how the rte tells
**  the client to stop.
*/
if (xact->branch_no < 0)
{
    tpfree((char *)xact);  /* Free the Tuxedo message buffer. */
    tpterm();      /* Perform Tuxedo client exit. */
    exit(0);
}

/*
**  Send a request to the Tuxedo server telling the server to perform
**  the TPC-A transaction.
*/
if (tpcall(servicename, (char *)xact, sizeof(struct xactinfo),
    (char **)&xact, (long *)&olen, TPSIGRSTRT) == -1)
{
    /*
    **  Got error from Tuxedo.
    */
    userlog("Client: tpcall() failed. Tuxedo error: %d\n", tperrno);

    /*
    **  If the error is 'timed out' increment the timed out counter.
    */
    if (tperrno == TPETIME)
        timeoutcount++;

    /*
    **  If the error is TPESVCFAIL or
    **     the error is 'timed out' and
    **     the timed out count is less than 3
    **  then return a non-fatal error to the rte.
    **  The non-fatal error is indicated by the -10 in the branch id
    **  position of the return message.
    */
    if (tperrno == TPESVCFAIL)
    {
        sprintf(outbuf, "-10,%05d,%09d,%010d,%011.0f,%s",
            xact->teller_no, xact->account_no, xact->amount,
            xact->balance, filler);
    }
    if (((tperrno == TPETIME) && (timeoutcount < 3)))
    {
        sprintf(outbuf, "-10,%05d,%09d,%010d,%011.0f,%s",
            xact->teller_no, xact->account_no, xact->amount,
            xact->balance, filler);
    }
    /*
    **  If we've timed out 3 or more times or
    **     the error wasn't **  'timed out' or TPESVCFAIL
    **  then return a fatal error.
    **  The fatal error is indicated by the -20 in the branch position
    **  of the return message.
    */
    else
    {
        sprintf(outbuf, "-20,%05d,%09d,%010d,%011.0f,%s",
            xact->teller_no, xact->account_no, xact->amount,
            xact->balance, filler);
    }
}
else
{
    /*
    **  Transaction completed successfully.  Put the branch id,
    **  teller id, account id, update amount, and new account balance
    **  into the return message.
    */
    sprintf(outbuf, "%04d,%05d,%09d,%010d,%011.0f,%s", xact->branch_no,
        xact->teller_no, xact->account_no, xact->amount,
        xact->balance, filler);
}

/*
**  Send message to the rte.  The client must send at least
**  200 bytes to the rte.
**  Clause 8.4.2: "The RTE ... Recieves 200 byte responses"
*/
outbuf[200] = '\0';
if (write(1, outbuf, 201) < 0)
{
    userlog("Client failed to write result to rte\n");
```

```
            tpfree((char *)xact);  /* Free the Tuxedo message buffer. */
            tpterm();      /* Perform Tuxedo client exit. */
            quit();
        }
    }
}
```

Tuxedo Server

t_srv.c

```
.#ifdef RCSID
static char *RCSid =
   "$Header: t_srv.c 7000000.5 93/08/06 14:14:45 bmoriart Generic<base> $ Copyr (c) 1993 Oracle";
#endif /* RCSID */

/*==================================================================+
 |           Copyright (c) 1992  Oracle Corp, Belmont, CA          |
 |                       All Rights Reserved                       |
 +==================================================================+
 | FILENAME
 |    t_srv.c
 | DESCRIPTION
 |    TPC-A Tuxedo server process.
 +==================================================================*/

#ifndef FALSE
# define FALSE 0
#endif
#ifndef TRUE
# define TRUE 1
#endif

#include <stdio.h>
#include <errno.h>
#include "atmi.h"
#include <sys/lock.h>


/*
**  The Tuxedo client and server processes communicate with the
**  xactinfo structure.  The client stores teller id, branch id, account id,
**  and update amount in the structure and sends it to the server.
**  The server stores the new account balance in the structure and sends
**  it back to the client.
*/
struct xactinfo
{
   long teller_no;
   long branch_no;
```

```
  long account_no;
  long amount;
  double balance;
};

struct xactinfo *xact;

long    branch_no;  /* Branch id.  Range: 1 to (1       * database scaling) */
long    teller_no;  /* Teller id.  Range: 1 to (10      * database scaling) */
long    account_no; /* Account id. Range: 1 to (100,000 * database scaling) */

long    amount;     /* Amount added to the balance */
              /* Clause 5.3.6: "The Delta amount field is a random */
              /* value within [-999999, +999999]" */

double  balance;      /* New balance of the account record */
               /* Clause 3.2.2: "Must be capable of representing */
               /* at least 10 significant decimal digits plus sign" */

char *  uid = "tpcb/tpcb"; /* Database user name and password */
int     retries = 0;  /* Discrete mode only: Number of retries. */
int     proc_no = 0;

int success_file = FALSE;  /* Write success file after every transaction */

/*
**  Function declarations.
*/

  /* TPC-A/B transaction functions */
extern int TPCinit();
extern int TPCexec();
extern int TPCexit();

  /* Durability test success file functions */
extern int tsuccinit();
extern int tsucclog();
extern int tsuccend();


tpsvrinit (argc, argv)
   int argc;
   char *argv[];
{
  /*
  **  Check for Optional Arguments.
  */
  if ((argc > 2) && (argv[argc-1][0] == 'd') && (argv[argc-1][1] == '\0')) {
    /*
    **  Durability test.  Write to success file after every
    **  transaction.
    */
    success_file = TRUE;
    argc--;
  }
  /*
  **  argv[last or next to last] -- process number of this process.
  */
  if (argc > 1) {
    proc_no = atoi (argv[argc-1]);
  }
  if (success_file) {
    if (tsuccinit(proc_no)<0) exit (1);
  }
  if (TPCinit(proc_no) == -1)
  {
     /* Error in TPCinit().  Write message to log file and exit. */
     userlog("Error from TPCinit()\n");
     exit(1);
  }
  userlog("Init completed\n");
}

void tpsvrdone ()
{
  /* Ignore return from TPCexit() */
  TPCexit();
  if (success_file)
  {
     if (tsuccend(proc_no))
       exit (1);
  }
}


TPCA1 (msg)
   TPSVCINFO *msg;
{
  /*
  **  Extract transaction information from the client's message structure.
  **  Store values in global variables used by TPCexec().
  */
  xact = (struct xactinfo *) msg->data;
  teller_no = xact->teller_no;          /* Teller record to update */
  branch_no = xact->branch_no;          /* Branch record to update */
  account_no = xact->account_no;        /* Account record to update */
  amount = xact->amount;                /* Amount of update */

  if (TPCexec() == -1)
  {
    /* Error in TPCexec().  Write message to log file and exit. */
    userlog("Error from TPCexec().  Exiting.\n");
    exit(1);
  }

  if (success_file)
  {
    if (tsucclog(proc_no,account_no,teller_no,branch_no,amount,balance))
      exit (1);
  }


  xact->balance = balance;   /* Return new account balance to client. */

  /* Return success to client. */
  tpreturn (TPSUCCESS, 0, (char *)xact, sizeof (struct xactinfo), 0);
}

TPCA2  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA3  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA4  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA5  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA6  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA7  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA8  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA9  (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA10 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA11 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA12 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA13 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA14 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA15 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA16 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA17 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA18 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA19 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA20 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA21 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA22 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA23 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA24 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA25 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA26 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA27 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA28 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA29 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA30 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA31 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA32 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA33 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
```

```
TPCA34 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA35 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA36 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA37 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA38 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA39 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA40 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA41 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA42 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA43 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA44 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA45 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA46 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA47 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA48 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA49 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
TPCA50 (msg) TPSVCINFO *msg; { return (TPCA1 (msg)); }
```

ab_trans.c:

```
/*================================================================+
 |          Copyright (c) 1992  Oracle Corp, Belmont, CA          |
 |                    UNIX PERFORMANCE GROUP                      |
 |                     All Rights Reserved                        |
 +================================================================*/

#include <stdio.h>

/*
** Global variables.
*/
extern long    account_no;   /* Account id to update */
extern long    branch_no;    /* Branch id to update */
extern long    teller_no;    /* Teller id to update */
extern long    amount;       /* Amount added to the balance */
extern double  balance;      /* New balance of the account record */
extern char *  uid;          /* Database user name and password */

extern int     retries;  /* Number of retries in the discrete transaction */

/*
**  Oracle variable type definitions.
*/

#define  SQLT_CHR  1          /* (ORANET TYPE) character string */
#define  SQLT_NUM  2          /* (ORANET TYPE) oracle numeric */
#define  SQLT_INT  3          /* (ORANET TYPE) integer */
#define  SQLT_FLT  4          /* (ORANET TYPE) Floating point number */
#define  SQLT_RID  11         /* rowid */
```

```
#define  SQLT_DAT  12                        /* date in oracle format */

/*
**  Oracle cursor structure.
*/
struct csrdef
{
   short        csrrc;                       /* return code */
   unsigned short csrft;                     /* function type */
   unsigned long  csrrpc;                   /* rows processed count */
   unsigned short csrpeo;                   /* parse error offset */
   unsigned char  csrfc;                     /* function code */
   unsigned char  csrfil;                      /* filler  */
   unsigned short csrarc;                    /* reserved, private */
   unsigned char  csrwrn;                     /* warning flags */
   unsigned char  csrflg;                     /* error flags */
   /*             *** Operating system dependent ***               */
   unsigned int   csrcn;                     /* cursor number */
   struct {                                /* rowid structure */
     struct {
        unsigned long   tidtrba;         /* rba of first blockof table */
        unsigned short  tidpid;           /* partition id of table */
        unsigned char   tidtbl;            /* table id of table */
        }             ridtid;
     unsigned long   ridbrba;                /* rba of datablock */
     unsigned short  ridsqn;         /* sequence number of row in block */
     } csrrid;
   unsigned int   csrose;               /* os dependent error code */
   unsigned char  csrchk;                       /* check byte */
   unsigned char  crsfill[26];            /* private, reserved fill */
};

typedef struct csrdef csrdef;
typedef struct csrdef ldadef;

void errrpt();

ldadef tpclda;
char   tpchda[256];

#define SQLTXT \
"\
begin\
  dbms_transaction.begin_discrete_transaction;\
  loop begin\
    update account\
      set account_balance = account_balance + :dlta\
      where account_id = :acct;\
    insert into history values\
      (:tell, :bran, :acct, :dlta, sysdate,\
     ’%05d-6789012345678901234567890123456789012345678’);\
    update teller\
      set teller_balance = teller_balance + :dlta\
      where teller_id = :tell;\
    update branch\
      set branch_balance = branch_balance + :dlta\
      where branch_id = :bran;\
    commit;\
    :bala := tpcab_pack.account_bal;\
    exit;\
    exception\
      when dbms_transaction.discrete_transaction_failed or \
         dbms_transaction.consistent_read_failure then\
         rollback;\
         :retr := :retr + 1;\
    end;\
  end loop;\
end;\
"

csrdef  * csr;          /* Cursor */




/*
**  TPCinit: perform database initialization.  Log on to the database.
**      Parse the transaction.  Bind the transaction variables.
**      Return 0 on success, -1 on failure.
*/
TPCinit(proc_no)
int proc_no;
{
    char sqlbuf[1024];

    /*
    ** Log on to the database
    */
    if (orlon(&tpclda, tpchda, uid, -1, (char *) -1, -1, 0))
    {
          errrpt(&tpclda);
          return -1;
    }

    if (ocicof(&tpclda))
    {
        errrpt(&tpclda);
        return -1;
    }
```

```
/* Allocate cursor */
csr  = (csrdef *)malloc(sizeof(csrdef));
if (csr == (csrdef *)0)
{
    fprintf(stderr, "Error: TPCinit(): 0 returned by malloc\n");
    return -1;
}

/* Open cursor */
if (ociope(csr, &tpclda, (char *)0, 0, -1, uid, -1))
{
    errrpt(csr);
    return -1;
}

sprintf(sqlbuf, SQLTXT, proc_no);

/* Parse sql statement */
if (osql3(csr, sqlbuf, -1))
{
    errrpt(csr);
    return -1;
}

/* Bind variables */

if (obndrv(csr, ":ACCT", -1, &account_no, sizeof(account_no), SQLT_INT,
    -1, (short *) -1, (char *) -1, -1, -1))
{
    errrpt(csr);
    return -1;
}

if (obndrv(csr, ":BALA", -1, &balance, sizeof(balance), SQLT_FLT,
    -1, (short *) -1, (char *) -1, -1, -1))
{
    errrpt(csr);
    return -1;
}

if (obndrv(csr, ":BRAN", -1, &branch_no, sizeof(branch_no), SQLT_INT,
    -1, (short *) -1, (char *) -1, -1, -1))
{
    errrpt(csr);
    return -1;
}

if (obndrv(csr, ":DLTA", -1, &amount, sizeof(amount), SQLT_INT,
    -1, (short *) -1, (char *) -1, -1, -1))
{
        errrpt(csr);
        return -1;
    }

    if (obndrv(csr, ":TELL", -1, &teller_no, sizeof(teller_no), SQLT_INT,
        -1, (short *) -1, (char *) -1, -1, -1))
    {
        errrpt(csr);
        return -1;
    }

    if (obndrv(csr, ":RETR", -1, &retries, sizeof(retries), SQLT_INT,
        -1, (short *) -1, (char *) -1, -1, -1))
    {
        errrpt(csr);
        return -1;
    }
    return 0;
}

/*
** TPCexec: Execute the transaction.
**     Return 0 on success, -1 on failure.
*/
TPCexec()
{
    char msg[2048];

    if (ociexe(csr))
    {
        if (csr->csrrc)
        {
            (void) ocierr(csr, csr->csrrc, msg, 2048);
            (void) fprintf(stderr, "%s\n", msg);
        }
        orol(&tpclda);
        return -1;
    }
    return 0;
}

/*
**  TPCexit: Close cursor and log off database.
**      Return 0 on success, -1 on failure.
*/
TPCexit()
{
    /* Close cursor */
    if (ociclo(csr))
        errrpt(csr);
    /* Free cursor */
    free(csr);

    /* Log off database */
    ocilof(&tpclda);

    return 0;
}
```

Appendix B: Database Definition and Load

File Definitions for ABTH Tables

The following Bourne shell scripts, “C” code and SQL scripts were used to define, create and load the Account, Teller, Branch and History tables.

The shell script that creates the database uses logical volumes for the Oracle data files. The mapping between logical volumes and the disk partitions are listed before the ABTH Table Sample Data later in Appendix B.

The following shell script calls Oracle’s SQL interpreter to handle various SQL statements. It creates the database.

cht_bld2200TPS.cluster

```
#===================================================================+
#            Copyright (c) 1993  Oracle Corp, Belmont, CA           |
#                       UNIX PERFORMANCE GROUP                      |
#                        All Rights Reserved                        |
#===================================================================+
BENCH_HOME=$ORACLE_HOME/bench
TPCAB_SQL=$BENCH_HOME/tpc/tpcab/sql
TPCAB_ADMIN=$BENCH_HOME/tpc/tpcab/admin
GEN_SQL=$BENCH_HOME/gen/sql

MULT=500

echo “creating datbase”
date

sqldba <<!
   set echo on
   connect internal
   startup  pfile=$TPCAB_ADMIN/p_create_cht.ora  nomount
   create database tpcb controlfile reuse
            maxdatafiles 200
       datafile ‘/tpc_db/orasys’  size  850M reuse
       logfile  ‘/tpc_db/log0’  size 900M reuse,
            ‘/tpc_db/log1’  size 900M reuse;

   exit
!

echo “done creating datbase”
date

sqldba <<!
```

```
    connect internal
   create rollback segment s1  storage (initial 100k minextents 2 next 10k);
   create rollback segment s2  storage (initial 100k minextents 2 next 10k);
   create rollback segment s3  storage (initial 100k minextents 2 next create
rollback segment s4  storage (initial 100k minextents 2 next 10k);
   create rollback segment s5  storage (initial 100k minextents 2 next 10k);
   create rollback segment s6  storage (initial 100k minextents 2 next 10k);
   create rollback segment s7  storage (initial 100k minextents 2 next 10k);
   create rollback segment s8  storage (initial 100k minextents 2 next 10k);
   create rollback segment s9  storage (initial 100k minextents 2 next 10k);
   create rollback segment s10 storage (initial 100k minextents 2 next 10k);
   create rollback segment s11  storage (initial 100k minextents 2 next 10k);
   create rollback segment s12  storage (initial 100k minextents 2 next 10k);
   create rollback segment s13  storage (initial 100k minextents 2 next 10k);
   create rollback segment s14  storage (initial 100k minextents 2 next 10k);
   create rollback segment s15  storage (initial 100k minextents 2 next 10k);
   create rollback segment s16  storage (initial 100k minextents 2 next 10k);
   create rollback segment s17  storage (initial 100k minextents 2 next 10k);
   create rollback segment s18  storage (initial 100k minextents 2 next 10k);
   create rollback segment s19  storage (initial 100k minextents 2 next 10k);
   create rollback segment s20  storage (initial 100k minextents 2 next 10k);
   create rollback segment s21  storage (initial 100k minextents 2 next 10k);
   create rollback segment s22  storage (initial 100k minextents 2 next 10k);
   create rollback segment s23  storage (initial 100k minextents 2 next 10k);
   create rollback segment s24  storage (initial 100k minextents 2 next 10k);
   create rollback segment s25  storage (initial 100k minextents 2 next 10k);
   create rollback segment s26  storage (initial 100k minextents 2 next 10k);
   create rollback segment s27  storage (initial 100k minextents 2 next 10k);
    shutdown
    connect internal
    startup pfile=$TPCAB_ADMIN/p_build_cht.ora
    exit
!

#
# Create acct tablespace which will hold account table
#
# Create hist tablespace to hold history table
#
#

echo “creating tablespaces”
date

sqldba lmode=y <<!
connect internal
    create tablespace acct datafile    ‘/tpc_db/acct0’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct1’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct2’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct3’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct4’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct5’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct6’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct7’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct8’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct9’   size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct10’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct11’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct12’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct13’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct14’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct15’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct16’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct17’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct18’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct19’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct20’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct21’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct22’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct23’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct24’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct25’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct26’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct27’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct28’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct29’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct30’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct31’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct32’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct33’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct34’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct35’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct36’  size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct37’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct38’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct39’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct40’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct41’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct42’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct43’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct44’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct45’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct46’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct47’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct48’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct49’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct50’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct51’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct52’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct53’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct54’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct55’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct56’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct57’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct58’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct59’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct60’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct61’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct62’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct63’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct64’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct65’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct66’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct67’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct68’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct69’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct70’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct71’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct72’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct73’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct74’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct75’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct76’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct77’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct78’ size 500M reuse;
    alter tablespace acct add datafile ‘/tpc_db/acct79’ size 500M reuse;

    shutdown
    connect internal
    startup pfile=$TPCAB_ADMIN/p_build_cht.ora
         exit;
 !

 sqldba lmode=y <<! > bld2200.cat.out
    set echo off;
    connect sys/change_on_install
    @?/rdbms/admin/catalog;
    @?/rdbms/admin/catexp;
    @?/rdbms/admin/catproc;
    exit;
 !


 echo “creating account table and cluster”
 date

 sqldba <<!
```

```
   CONNECT system/manager;
   GRANT CONNECT,RESOURCE,UNLIMITED TABLESPACE TO tpcb IDENTIFIED BY
tpcb;
   CONNECT tpcb/tpcb;
   DROP CLUSTER acluster INCLUDING TABLES;
   CREATE CLUSTER acluster
   (
      account_id number(10,0)
   )
   HASHKEYS   220000000
   HASH IS    account_id
   SIZE       138
   INITRANS   2
   PCTFREE    0
   TABLESPACE acct
   STORAGE
   (
      INITIAL     400M
      NEXT        400M
      PCTINCREASE 0
      MINEXTENTS  80
   );
   CREATE TABLE account
   (
      account_id        NUMBER(10,0),
      branch_id         NUMBER,
      account_balance   NUMBER,
      filler            VARCHAR2(97)
   )
  CLUSTER acluster(account_id);
  EXIT;
!
date
```

The following is a shell script that calls Oracle’s SQL interpreter to handle various SQL statements. It fills the database with initial data, creates tables used by the benchmark, and creates a trigger used interally by Oracle.

cht_bld2200TPS.continue

```
BENCH_HOME=$ORACLE_HOME/bench
TPCAB_SQL=$BENCH_HOME/tpc/tpcab/sql
TPCAB_ADMIN=$BENCH_HOME/tpc/tpcab/admin
GEN_SQL=$BENCH_HOME/gen/sql

MULT=500

echo “loading account”
date

runon  0 ab_load a 5000000        1 > load0.out &
runon  1 ab_load a 5000000  5000001 > load1.out &
runon  2 ab_load a 5000000 10000001 > load2.out &
runon  3 ab_load a 5000000 15000001 > load3.out &
runon  4 ab_load a 5000000 20000001 > load4.out &
runon  5 ab_load a 5000000 25000001 > load5.out &
runon  6 ab_load a 5000000 30000001 > load6.out &
runon  7 ab_load a 5000000 35000001 > load7.out &
runon  8 ab_load a 5000000 40000001 > load8.out &
runon  9 ab_load a 5000000 45000001 > load9.out &

runon 10 ab_load a 5000000 50000001 > loada.out &
runon 11 ab_load a 5000000 55000001 > loadb.out &
runon 12 ab_load a 5000000 60000001 > loadc.out &
runon 13 ab_load a 5000000 65000001 > loadd.out &
runon 14 ab_load a 5000000 70000001 > loade.out &
runon 15 ab_load a 5000000 75000001 > loadf.out &
runon 16 ab_load a 5000000 80000001 > loadg.out &
runon  1 ab_load a 5000000 85000001 > loadh.out &
runon  2 ab_load a 5000000 90000001 > loadi.out &
runon  3 ab_load a 5000000 95000001 > loadj.out &

runon  4 ab_load a 5000000 100000001 > loadk.out &
runon  5 ab_load a 5000000 105000001 > loadl.out &
runon  6 ab_load a 5000000 110000001 > loadm.out &
runon  7 ab_load a 5000000 115000001 > loadn.out &
runon  8 ab_load a 5000000 120000001 > loado.out &
runon  9 ab_load a 5000000 125000001 > loadp.out &
runon 10 ab_load a 5000000 130000001 > loadq.out &
wait
runon 12 ab_load a 5000000 135000001 > loadr.out &
runon 13 ab_load a 5000000 140000001 > loads.out &
runon 14 ab_load a 5000000 145000001 > loadt.out &

runon 15 ab_load a 5000000 150000001 > loadu.out &
runon 16 ab_load a 5000000 155000001 > loadv.out &
runon  0 ab_load a 5000000 160000001 > loadw.out &
runon  1 ab_load a 5000000 165000001 > loadx.out &
runon  2 ab_load a 5000000 170000001 > loady.out &
runon  3 ab_load a 5000000 175000001 > loadz.out &
runon  4 ab_load a 5000000 180000001 > loadA.out &
runon  5 ab_load a 5000000 185000001 > loadB.out &
runon  6 ab_load a 5000000 190000001 > loadC.out &
runon  7 ab_load a 5000000 195000001 > loadD.out &

runon  8 ab_load a 5000000 200000001 > loadE.out &
runon  9 ab_load a 5000000 205000001 > loadF.out &
runon 10 ab_load a 5000000 210000001 > loadG.out &
runon 11 ab_load a 5000000 215000001 > loadH.out &
wait

echo “done loading account”
date

echo “building teller and branch tables”
date
sqldba <<!

   CONNECT tpcb/tpcb

   CREATE TABLE teller (
      teller_id         NUMBER(10,0),
      branch_id         NUMBER(10,0),
      teller_balance    NUMBER(10,0),
      filler            CHAR(97)
   )
      PCTFREE 40
      PCTUSED 4
      STORAGE ( initial 210K next 210K pctincrease 0 minextents 48 );

   CREATE TABLE branch
   (
      branch_id         NUMBER,
      branch_balance    NUMBER,
      filler            CHAR(98)
   )
   PCTFREE 90
   PCTUSED 4
   STORAGE (initial 40K next 40K pctincrease 0 minextents 121 );

    EXIT;
!


runon 1 ab_load t 22000
runon 1 ab_load b 2200
echo  “done loading branch and teller”
echo  “begin rollback segment creation”


#  Create tables for processing benchmark results
```

```
#
sqlplus sys/change_on_install @$GEN_SQL/orst_cre
sqlplus sys/change_on_install @$TPCAB_SQL/ab_stat
sqlplus sys/change_on_install @$GEN_SQL/pst_c

##
##  Create trigger
##
sqldba <<!
    connect tpcb/tpcb
    @$TPCAB_SQL/ab_trig
    exit
!
```

The following shell script calls Oracle’s SQL interpreter to handle various SQL statements. It allocates space for the history file.

cht.history.alloc

```
#!/bin/sh
date
sqldba lmode=y << !
   connect internal;
        drop tablespace hist0 including contents;
        drop tablespace hist1 including contents;
        drop tablespace hist2 including contents;
        drop tablespace hist3 including contents;
        drop tablespace hist4 including contents;
        drop tablespace hist5 including contents;
        drop tablespace hist6 including contents;
   drop tablespace histalloc including contents;
   create tablespace histalloc     datafile ‘/tpc_db/multhist0’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist1’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist2’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist3’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist4’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist5’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist6’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist7’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist8’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist9’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist10’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist11’  size 90M reuse;
    alter tablespace histalloc add datafile ‘/tpc_db/multhist12’  size 90M reuse;
exit;
```

The following C source sode is executed by cht_bld2200TPS.continue to fill in the database.

ab_load.c:

```
/*===================================================================+
 |  Copyright (c) 1992  Oracle Corp, Belmont, CA         |
 | All Rights Reserved |
 +===================================================================+
 | FILENAME
 |  ab_load.c
 | DESCRIPTION
 |  load database tables for TPC-A or -B benchmark.
 |
 +===================================================================*/

typedef char  b1;
typedef short b2;
typedef int   b4;

typedef unsigned char  ub1;
typedef unsigned short ub2;
typedef unsigned int   ub4;

typedef ub1 text;
```

```
#include <stdio.h>

/* input data types */
#define  SQLT_CHR  1        /* (ORANET TYPE) character string */
#define  SQLT_INT  3        /* (ORANET TYPE) integer */

/*
**  Oracle cursor structure.
*/
struct csrdef
{
   short          csrrc;     /* return code */
   unsigned short csrft;   /* function type */
   unsigned long  csrrpc;/* rows processed count */
   unsigned short csrpeo;  /* parse error offset */
   unsigned char  csrfc;   /* function code */
   unsigned char  csrfil;        /* filler  */
   unsigned short csrarc;   /* reserved, private */
   unsigned char  csrwrn;  /* warning flags */
   unsigned char  csrflg;    /* error flags */
   /*     *** Operating system dependent ***       */
   unsigned int   csrcn;   /* cursor number */
   struct {   /* rowid structure */
     struct {
        unsigned long   tidtrba;/* rba of first blockof table */
        unsigned short  tidpid;      /* partition id of table */
        unsigned char   tidtbl;          /* table id of table */
        }    ridtid;
     unsigned long   ridbrba;/* rba of datablock */
     unsigned short  ridsqn;          /* sequence number of row in block
*/
     } csrrid;
   unsigned int   csrose;          /* os dependent error code */
   unsigned char  csrchk;     /* check byte */
   unsigned char  crsfill[26];      /* private, reserved fill */
};

typedef struct csrdef csrdef;
typedef struct csrdef ldadef;

ldadef tpclda;
char   tpchda[256];

/* SQL statements */

#define SQLTXT_ACCT \
“INSERT INTO account(account_id, account_balance, branch_id, filler)\
 VALUES (:1, :2, :3, :4)”

#define SQLTXT_TELLER \
“INSERT INTO teller(teller_id, teller_balance, branch_id, filler)\
 VALUES (:1, :2, :3, :4)”

#define SQLTXT_BRANCH \
“INSERT INTO branch(branch_id, branch_balance, filler)\
 VALUES (:1, :2, :3)”

/* SQL cursor */

csrdef * csr;

#define BRANCH  1   /* Table IDs; command line arg mapped here */
#define TELLER  2
#define ACCOUNT 3

#define LOOP  100 /* Number of rows to insert before committing. */

#define INITBAL -1111111111     /* Init balance. Use all 10 digits */

#define PAD97 \
“1234567890123456789012345678901234567890123456789\
12345678901234567890123456789012345678901234567”

#define PAD98 \
“12345678901234567890123456789012345678901234567890\
123456789012345678901234567890123456789012345678”

#define MIN(a,b) ((a) < (b) ? (a) : (b))

/*==================================================================+
 | ROUTINE NAME
 |  main
 | DESCRIPTION
 |  main routine
 | ARGUMENTS
 |  tpcbload <tablename> <#_rows_to_insert> [#_row_to_start]
 +==================================================================*/

main(argc, argv)
    int argc;
    char *argv[];
{
    char   * uid = “tpcb/tpcb”;
    char   * upasswd = “tpcb”;
    int     tellbran;  /* branch to which teller belongs */
    int     acctbran;  /* branch to which account belongs */
    int     init_bal;
    char    rowpad[128];
    int     loop;/* for array inserts    */
    char    sqlbuf[256];
    int     i, j;
    int     which_table = 0;    /* 1=acct, 2=teller, 3=branch   */
    long    nrows;          /* # of rows to insert      */
    long    row; /* row/key-value counter    */
    long    start;          /* starting key value       */
    long    end; /* ending key value     */
    int     loopcount;      /* insert loop counter      */
    int     err = 0;

    void errrpt();
    double begin_time, end_time;
    double begin_cpu, end_cpu;
    static double gettime(), getcpu();

    /*
    **  Parse command line -- look for specific table to load.
    */

    if (argc < 3) usage();

    /*
    **  argv[1]:  table name.
    */

    switch (argv[1][0])
    {
        case ‘a’:/* account table    */
    which_table = ACCOUNT;
    break;
        case ‘t’:/* teller table     */
    which_table = TELLER;
    break;
        case ‘b’:/* branch table     */
    which_table = BRANCH;
    break;
        default:
    usage();
    break;
    }

    /*
    **  argv[2]:  # of rows to insert.
    */
```

```
  if ((nrows = atoi(argv[2])) < 1 )
  {
    fprintf(stderr, “Invalid number of rows to insert:  ‘%d’\n”,
nrows );
    usage();
  }

  /*
  ** argv[3]:  starting row #  (optional).
  */

  if (argc > 3)
  {
    if ((start = atoi(argv[3])) < 1 )
    {
  fprintf(stderr, “Invalid start offset:  ‘%d’\n”, start );
  exit();
    }
  }
  else start = 1;
    end = start + nrows - 1;

  /*
  ** Log on to the database
  */
  if (orlon(&tpclda, tpchda, uid, -1, (char *) -1, -1, 0))

  {
     errrpt(&tpclda);
     return -1;
  }

  if (ocicof(&tpclda))
  {
    errrpt(&tpclda);
    return -1;
  }

  csr = (csrdef *)malloc(sizeof(csrdef));

  if (csr == (csrdef *)0)
  {
    fprintf(stderr, “Error: 0 returned by malloc\n”);
    exit(-1);
  }

  if (ociope(csr, &tpclda, (char *)0, 0, -1, uid, -1))
    errrpt(csr);

  /* prepare the account insert cursor */
  if (which_table == ACCOUNT)
  {
    sprintf(sqlbuf, SQLTXT_ACCT);

    init_bal = INITBAL;

    strcpy(rowpad, PAD97);

    if (osql3(csr, sqlbuf, -1))
  errrpt(csr);

    if (obndrn(csr, 1, &row, sizeof(row), SQLT_INT, -1,
  (short *)NULL, -1))
  errrpt(csr);

    if (obndrn(csr, 2, &init_bal, sizeof(init_bal), SQLT_INT, -1,
  (short *)NULL, -1))
  errrpt(csr);

    if (obndrn(csr, 3, &acctbran, sizeof(acctbran), SQLT_INT, -1,
  (short *)NULL, -1))
  errrpt(csr);
```

```
        if (obndrn(csr, 4, rowpad, strlen(rowpad), SQLT_CHR, -1,
    (short *)NULL, -1))
    errrpt(csr);

        printf(“Loading ACCOUNT table with %d rows starting with %d
...\n   “,
    nrows, start );
    }

    if (which_table == TELLER)
    {
        sprintf(sqlbuf, SQLTXT_TELLER);
        init_bal = INITBAL;
        strcpy(rowpad, PAD97);

        if (osql3(csr, sqlbuf, -1))
    errrpt(csr);

        if (obndrn(csr, 1, &row, sizeof(row), SQLT_INT, -1,
    (short *)NULL, -1))
    errrpt(csr);

        if (obndrn(csr, 2, &init_bal, sizeof(init_bal), SQLT_INT, -1,
    (short *)NULL, -1))
    errrpt(csr);

        if (obndrn(csr, 3, &tellbran, sizeof(tellbran), SQLT_INT, -1,
    (short *)NULL, -1))
    errrpt(csr);

        if (obndrn(csr, 4, rowpad, strlen(rowpad), SQLT_CHR, -1,
    (short *)NULL, -1))
    errrpt(csr);

        printf(“Loading TELLER table with %d rows starting with %d
...\n   “,
    nrows, start );
    }

    if (which_table == BRANCH)
    {
        sprintf(sqlbuf, SQLTXT_BRANCH);
        init_bal = INITBAL;
        strcpy(rowpad, PAD98);

        if (osql3(csr, sqlbuf, -1))
    errrpt(csr);

        if (obndrn(csr, 1, &row, sizeof(row), SQLT_INT, -1,
    (short *)NULL, -1))
    errrpt(csr);

        if (obndrn(csr, 2, &init_bal, sizeof(init_bal), SQLT_INT, -1,
    (short *)NULL, -1))
    errrpt(csr);

        if (obndrn(csr, 3, rowpad, strlen(rowpad), SQLT_CHR, -1,
    (short *)NULL, -1))
    errrpt(csr);

        printf(“Loading TELLER table with %d rows starting with %d
...\n   “,
    nrows, start );
    }

    begin_time = gettime();
    begin_cpu = getcpu();

    loopcount = 0;
    row = start;
    while (row <= end)
    {
        loop = MIN(LOOP, end - row + 1);

        for (i = 0; i < loop; i++, row++)
        {
    acctbran = ((row - 1) / 100000) + 1;
    tellbran = ((row - 1) / 10) + 1;

    if (err = oexec(csr))
    {
        orol(&tpclda);
        errrpt(csr);
    }
        }

        if (err = ocom(&tpclda))
        {
    orol(&tpclda);
    errrpt(&tpclda);
        }

        if ((++loopcount) % 50)
    printf( “.” );
        else
    printf( “ row %d committed.\n   “, row - 1);
    }

    end_time = gettime();
    end_cpu = getcpu();

    printf( “Load completed.  %d records processed.\n”, nrows );
    printf( “      in %10.2f real, %10.2f cpu.\n”,
        end_time-begin_time, end_cpu-begin_cpu );

    if (ociclo(csr))
        errrpt(csr);

    free(csr);

    ocilof(&tpclda);

    exit(0);

}

usage()
{
    printf(“\n”);
    printf(
        “Usage:  tpcbload <table_name> <#_rows_to_insert>
[starting_row_#]\n”);
    printf(“\n”);
    exit(1);
}

void errrpt(cur)
    csrdef  *cur;
{
    char msg[2048];

    if (cur->csrrc)
    {
         (void) ocierr(cur, cur->csrrc, msg, 2048);
         (void) fprintf(stderr, “%s\n”, msg);
    }
    exit(0);
}
```

This is an input file to Oracle's SQL*PLUS program. It recreates the history file before each benchmark run.

ab_hist.sql

```
!
date
tpcb/tpcb

rem
rem ==================================================================+
rem            Copyright (c) 1991  Oracle Corp, Belmont, CA         |
rem                        All Rights Reserved                       |
rem ==================================================================+
rem  FILENAME
rem       ab_hist.sql
rem  DESCRIPTION
rem ==================================================================*/
rem


    DROP TABLE history;
    DROP VIEW history;

rem the following will fail
    create table history_coalesce (x number)
         tablespace hist
         storage (initial 2000M);


    CREATE TABLE history
    (
        teller_id         NUMBER,
        branch_id         NUMBER,
        account_id        NUMBER,
        amount            NUMBER,
        timestamp         DATE,
        filler            VARCHAR2(39)
    )
    tablespace histalloc
    storage (initial 4k
                  minextents 1
                  pctincrease 0
                  freelist groups 13
                  freelists 17
                  ) pctfree 0;
    alter table history allocate extent (size 88M freelist group 1);
    alter table history allocate extent (size 88M freelist group 2);
    alter table history allocate extent (size 88M freelist group 3);
    alter table history allocate extent (size 88M freelist group 4);
    alter table history allocate extent (size 88M freelist group 5);
    alter table history allocate extent (size 88M freelist group 6);
    alter table history allocate extent (size 88M freelist group 7);
    alter table history allocate extent (size 88M freelist group 8);
    alter table history allocate extent (size 88M freelist group 9);
    alter table history allocate extent (size 88M freelist group 10);
    alter table history allocate extent (size 88M freelist group 11);
    alter table history allocate extent (size 88M freelist group 12);
    alter table history allocate extent (size 88M freelist group 13);

    EXIT;
```

ab_trig.sql

```
rem
rem ==================================================================+
rem            Copyright (c) 1992  Oracle Corp, Belmont, CA         |
rem                       UNIX PERFORMANCE GROUP                     |
rem                        All Rights Reserved                       |
rem ==================================================================+
rem  FILENAME
rem       ab_trig.sql
rem  DESCRIPTION
rem       Create Trigger & Package for TPC A&B
rem =================================================================*/
rem
drop package tpcab_pack;

create package tpcab_pack is
  account_bal number;
  end;
/

drop trigger tpcab_trig;

create trigger tpcab_trig after update on account for each row
begin tpcab_pack.account_bal := :new.account_balance; end;
/
```

An Oracle data or log file can be an Irix Logical Volume which is striped over two disk partitions.

The following table shows the mapping between Oracle data or log file (and its table space), Irix Logical Volume, and Irix disk partitions.

Partition	Type	Start Blk	End Blk	Size	Use	
				(MB)	Table	
					Space	Data File
dks110d1s12	lv3.1	4128	692127	336	acct	acct1
dks110d2s12	lv14.0	4128	692127	336	acct	acct14
dks110d3s12	lv25.0	4128	692127	336	acct	acct25
dks110d4s12	lv35.1	4128	692127	336	acct	acct35
dks110d5s12	lv46.0	4128	692127	336	acct	acct46
dks110d6s12	lv56.1	4128	692127	336	acct	acct56
dks110d7s12	lv67.0	4128	692127	336	acct	acct67
dks110d8s12	lv77.1	4128	692127	336	acct	acct77
dks111d1s12	lv3.0	4128	692127	336	acct	acct3
dks111d2s12	lv13.1	4128	692127	336	acct	acct13
dks111d3s12	lv24.1	4128	692127	336	acct	acct24
dks111d4s12	lv35.0	4128	692127	336	acct	acct35
dks111d5s12	lv45.1	4128	692127	336	acct	acct45
dks111d6s12	lv56.0	4128	692127	336	acct	acct56
dks111d7s12	lv66.1	4128	692127	336	acct	acct66
dks111d8s12	lv77.0	4128	692127	336	acct	acct77
dks113d1s12	lv2.0	4128	692127	336	acct	acct2
dks113d2s12	lv12.1	4128	692127	336	acct	acct12
dks113d3s12	lv23.1	4128	692127	336	acct	acct23
dks113d4s12	lv34.0	4128	692127	336	acct	acct34
dks113d5s12	lv44.1	4128	692127	336	acct	acct44
dks113d6s12	lv55.0	4128	692127	336	acct	acct55
dks113d7s12	lv65.1	4128	692127	336	acct	acct65
dks113d8s12	lv76.0	4128	692127	336	acct	acct76
dks114d1s12	lv1.1	4128	692127	336	acct	acct1
dks114d2s12	lv12.0	4128	692127	336	acct	acct12
dks114d3s12	lv23.0	4128	692127	336	acct	acct23
dks114d4s12	lv33.1	4128	692127	336	acct	acct33
dks114d5s12	lv44.0	4128	692127	336	acct	acct44
dks114d6s12	lv54.1	4128	692127	336	acct	acct54
dks114d7s12	lv65.0	4128	692127	336	acct	acct65
dks114d8s12	lv75.1	4128	692127	336	acct	acct75
dks115d1s12	lv1.0	4128	692127	336	acct	acct1
dks115d2s12	lv11.1	4128	692127	336	acct	acct11
dks115d3s12	lv22.1	4128	692127	336	acct	acct22
dks115d4s12	lv33.0	4128	692127	336	acct	acct33
dks115d5s12	lv43.1	4128	692127	336	acct	acct43
dks115d6s12	lv54.0	4128	692127	336	acct	acct54
dks115d7s12	lv64.1	4128	692127	336	acct	acct64
dks115d8s12	lv75.0	4128	692127	336	acct	acct75
dks116d1s12	lv0.1	4128	692127	336	acct	acct0
dks116d2s12	lv11.0	4128	692127	336	acct	acct11

```
dks116d3s12  lv22.0      4128    692127       336  acct  acct22
dks116d4s12  lv32.1      4128    692127       336  acct  acct32
dks116d5s12  lv43.0      4128    692127       336  acct  acct43
dks116d6s12  lv53.1      4128    692127       336  acct  acct53
dks116d7s12  lv64.0      4128    692127       336  acct  acct64
dks116d8s12  lv74.1      4128    692127       336  acct  acct74
dks116d9s7   lv80.0      4128   3932607    1918.2        log0.lv
dks116d10s7  lv81.0      4128   3932607    1918.2        log1.lv
dks116d11s7  lv82.0      4128   3932607    1918.2        mirror_log0.lv
dks116d12s7  lv83.0      4128   3932607    1918.2        mirror_log1.lv
dks117d1s12   lv0.0      4128    692127       336  acct  acct0
dks117d2s12  lv10.1      4128    692127       336  acct  acct10
dks117d3s12  lv21.1      4128    692127       336  acct  acct21
dks117d4s12  lv32.0      4128    692127       336  acct  acct32
dks117d5s12  lv42.1      4128    692127       336  acct  acct42
dks117d6s12  lv53.0      4128    692127       336  acct  acct53
dks117d7s12  lv63.1      4128    692127       336  acct  acct63
dks117d8s12  lv74.0      4128    692127       336  acct  acct74
dks117d9s7   lv80.1      4128   3932607    1918.2        log0.lv
dks117d10s7  lv81.1      4128   3932607    1918.2        log1.lv
dks117d11s7  lv82.1      4128   3932607    1918.2        mirror_log0.lv
dks117d12s7  lv83.1      4128   3932607    1918.2        mirror_log1.lv
dks1d2s12    lv21.0      4128    692127       336  acct  acct21
dks1d3s12    lv31.1      4128    692127       336  acct  acct31
dks2d1s12    lv10.0      4128    692127       336  acct  acct10
dks2d2s12    lv20.1      4128    692127       336  acct  acct20
dks2d4s12    lv42.0      4128    692127       336  acct  acct42
dks2d5s12    lv52.1      4128    692127       336  acct  acct52
dks2d6s12    lv63.0      4128    692127       336  acct  acct63
dks2d7s12    lv73.1      4128    692127       336  acct  acct73
dks3d1s12     lv9.1      4128    692127       336  acct  acct9
dks3d2s12    lv20.0      4128    692127       336  acct  acct20
dks3d3s12    lv31.0      4128    692127       336  acct  acct31
dks3d4s12    lv41.1      4128    692127       336  acct  acct41
dks3d5s12    lv52.0      4128    692127       336  acct  acct52
dks3d6s12    lv62.1      4128    692127       336  acct  acct62
dks3d7s12    lv73.0      4128    692127       336  acct  acct73
dks4d1s12     lv9.0      4128    692127       336  acct  acct9
dks4d2s12    lv19.1      4128    692127       336  acct  acct19
dks4d3s12    lv30.1      4128    692127       336  acct  acct30
dks4d4s12    lv41.0      4128    692127       336  acct  acct41
dks4d5s12    lv51.1      4128    692127       336  acct  acct51
dks4d6s12    lv62.0      4128    692127       336  acct  acct62
dks4d7s12    lv72.1      4128    692127       336  acct  acct72
dks5d1s12     lv8.1      4128    692127       336  acct  acct8
dks5d2s12    lv19.0      4128    692127       336  acct  acct19
dks5d3s12    lv30.0      4128    692127       336  acct  acct30
dks5d4s12    lv40.1      4128    692127       336  acct  acct40
dks5d5s12    lv51.0      4128    692127       336  acct  acct51
dks5d6s12    lv61.1      4128    692127       336  acct  acct61
dks5d7s12    lv72.0      4128    692127       336  acct  acct72
dks6d1s12     lv8.0      4128    692127       336  acct  acct8
dks6d2s12    lv18.1      4128    692127       336  acct  acct18
dks6d3s12    lv29.1      4128    692127       336  acct  acct29
dks6d4s12    lv40.0      4128    692127       336  acct  acct40
dks6d5s12    lv50.1      4128    692127       336  acct  acct50
dks6d6s12    lv61.0      4128    692127       336  acct  acct61
dks6d7s12    lv71.1      4128    692127       336  acct  acct71
dks70d1s12    lv2.1      4128    692127       336  acct  acct2
dks70d2s12   lv13.0      4128    692127       336  acct  acct13
dks70d3s12   lv24.0      4128    692127       336  acct  acct24
dks70d4s12   lv34.1      4128    692127       336  acct  acct34
dks70d5s12   lv45.0      4128    692127       336  acct  acct45
dks70d6s12   lv55.1      4128    692127       336  acct  acct55
dks70d7s12   lv66.0      4128    692127       336  acct  acct66
dks70d8s12   lv76.1      4128    692127       336  acct  acct76
dks71d1s12    lv7.0      4128    692127       336  acct  acct7
dks71d2s12   lv17.1      4128    692127       336  acct  acct17
dks71d3s12   lv28.1      4128    692127       336  acct  acct28
dks71d4s12   lv39.0      4128    692127       336  acct  acct39
dks71d5s12   lv49.1      4128    692127       336  acct  acct49
dks71d6s12   lv60.0      4128    692127       336  acct  acct60
```

```
dks71d7s12   lv70.1      4128    692127       336  acct  acct70
dks72d1s12    lv6.1      4128    692127       336  acct  acct6
dks72d2s12   lv17.0      4128    692127       336  acct  acct17
dks72d3s12   lv28.0      4128    692127       336  acct  acct28
dks72d4s12   lv38.1      4128    692127       336  acct  acct38
dks72d5s12   lv49.0      4128    692127       336  acct  acct49
dks72d6s12   lv59.1      4128    692127       336  acct  acct59
dks72d7s12   lv70.0      4128    692127       336  acct  acct70
dks73d1s12    lv6.0      4128    692127       336  acct  acct6
dks73d2s12   lv16.1      4128    692127       336  acct  acct16
dks73d3s12   lv27.1      4128    692127       336  acct  acct27
dks73d4s12   lv38.0      4128    692127       336  acct  acct38
dks73d5s12   lv48.1      4128    692127       336  acct  acct48
dks73d6s12   lv59.0      4128    692127       336  acct  acct59
dks73d7s12   lv69.1      4128    692127       336  acct  acct69
dks74d1s12    lv5.1      4128    692127       336  acct  acct5
dks74d2s12   lv16.0      4128    692127       336  acct  acct16
dks74d3s12   lv27.0      4128    692127       336  acct  acct27
dks74d4s12   lv37.1      4128    692127       336  acct  acct37
dks74d5s12   lv48.0      4128    692127       336  acct  acct48
dks74d6s12   lv58.1      4128    692127       336  acct  acct58
dks74d7s12   lv69.0      4128    692127       336  acct  acct69
dks74d8s12   lv79.1      4128    692127       336  acct  acct79
dks75d1s12    lv5.0      4128    692127       336  acct  acct5
dks75d2s12   lv15.1      4128    692127       336  acct  acct15
dks75d3s12   lv26.1      4128    692127       336  acct  acct26
dks75d4s12   lv37.0      4128    692127       336  acct  acct37
dks75d5s12   lv47.1      4128    692127       336  acct  acct47
dks75d6s12   lv58.0      4128    692127       336  acct  acct58
dks75d7s12   lv68.1      4128    692127       336  acct  acct68
dks75d8s12   lv79.0      4128    692127       336  acct  acct79
dks76d1s12    lv4.1      4128    692127       336  acct  acct4
dks76d2s12   lv15.0      4128    692127       336  acct  acct15
dks76d3s12   lv26.0      4128    692127       336  acct  acct26
dks76d4s12   lv36.1      4128    692127       336  acct  acct36
dks76d5s12   lv47.0      4128    692127       336  acct  acct47
dks76d6s12   lv57.1      4128    692127       336  acct  acct57
dks76d7s12   lv68.0      4128    692127       336  acct  acct68
dks76d8s12   lv78.1      4128    692127       336  acct  acct78
dks77d1s12    lv4.0      4128    692127       336  acct  acct4
dks77d2s12   lv14.1      4128    692127       336  acct  acct14
dks77d3s12   lv25.1      4128    692127       336  acct  acct25
dks77d4s12   lv36.0      4128    692127       336  acct  acct36
dks77d5s12   lv46.1      4128    692127       336  acct  acct46
dks77d6s12   lv57.0      4128    692127       336  acct  acct57
dks77d7s12   lv67.1      4128    692127       336  acct  acct67
dks77d8s12   lv78.0      4128    692127       336  acct  acct78
dks7d1s12     lv7.1      4128    692127       336  acct  acct7
dks7d2s12    lv18.0      4128    692127       336  acct  acct18
dks7d3s12    lv29.0      4128    692127       336  acct  acct29
dks7d4s12    lv39.1      4128    692127       336  acct  acct39
dks7d5s12    lv50.0      4128    692127       336  acct  acct50
dks7d6s12    lv60.1      4128    692127       336  acct  acct60
dks7d7s12    lv71.0      4128    692127       336  acct  acct71
```

ABTH Table Sample Data

```
SQL> describe account
 Name                            Null?    Type
 ------------------------------- -------- ----
 ACCOUNT_ID                               NUMBER(10)
 BRANCH_ID                                NUMBER
 ACCOUNT_BALANCE                          NUMBER
 FILLER                                   VARCHAR2(97)

SQL> describe branch
 Name                            Null?    Type
 ------------------------------- -------- ----
 BRANCH_ID                                NUMBER
 BRANCH_BALANCE                           NUMBER
 FILLER                                   CHAR(98)
```

```
SQL> describe teller
 Name                            Null?    Type
 ------------------------------- -------- ----
 TELLER_ID                                NUMBER(10)
 BRANCH_ID                                NUMBER(10)
 TELLER_BALANCE                           NUMBER(10)
 FILLER                                   CHAR(97)

SQL> describe history
 Name                            Null?    Type
 ------------------------------- -------- ----
 TELLER_ID                                NUMBER
 BRANCH_ID                                NUMBER
 ACCOUNT_ID                               NUMBER
 AMOUNT                                   NUMBER
 TIMESTAMP                                DATE
 FILLER                                   VARCHAR2(39)

SQL> set pagesize 500
SQL> select * from account where account_id < 15;

ACCOUNT_ID  BRANCH_ID ACCOUNT_BALANCE
---------- ---------- ---------------
FILLER
-------------------------------------------------------------------------
-------
        1         1      -1.110E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        2         1      -1.112E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        3         1      -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        4         1      -1.110E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        5         1      -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        6         1      -1.109E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        7         1      -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        8         1      -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        9         1      -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       10         1      -1.113E+09
```

```
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       11         1     -1.112E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       12         1     -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       13         1     -1.111E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       14         1     -1.112E+09
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567


14 rows selected.

SQL> select * from teller where teller_id < 15;

 TELLER_ID  BRANCH_ID TELLER_BALANCE
---------- ---------- --------------
FILLER
-------------------------------------------------------------------------
-------
        1         1       -723213
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        2         1      -1952190
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        3         1       2538121
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        4         1       -817169
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        5         1       -193256
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        6         1      -1832126
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        7         1       2133615
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

        8         1       1621276
1234567890123456789012345678901234567890123456789012345678901234567890123
```

```
4567890
12345678901234567

        9         1       1170073
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       10         1       2115744
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       11         2       -100531
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       12         2      -1333344
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       13         2       3929180
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567

       14         2       1221545
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
12345678901234567


14 rows selected.

SQL> select * from branch where branch_id < 15;

 BRANCH_ID BRANCH_BALANCE
---------- --------------
FILLER
-------------------------------------------------------------------------
-------
        1       4060875
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        2       2809800
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        3       4339114
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        4      -2295079
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        5     -11583540
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        6         80098
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678
```

```
        7      -1713340
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        8       4348028
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

        9       3095917
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

       10      -5262051
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

       11       4630804
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

       12      -1648514
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

       13       -823584
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678

       14       5270204
1234567890123456789012345678901234567890123456789012345678901234567890123
4567890
123456789012345678


14 rows selected.

SQL> select * from history where rownum < 15;

 TELLER_ID  BRANCH_ID ACCOUNT_ID     AMOUNT TIMESTAMP
---------- ---------- ---------- ---------- ---------
FILLER
---------------------------------------
       700         70    6981710      40612 13-APR-94
00001-678901234567890123456789012345678

        12          2     173008     734149 13-APR-94
00001-678901234567890123456789012345678

        14          2  145200374    -367215 13-APR-94
00001-678901234567890123456789012345678

       734         74  206199128    -463190 13-APR-94
00001-678901234567890123456789012345678

       722         73    7202599    -148774 13-APR-94
00001-678901234567890123456789012345678

        36          4     342972    -618940 13-APR-94
00001-678901234567890123456789012345678

       715         72    7149143    -249049 13-APR-94
00001-678901234567890123456789012345678
```

```
      732        74  28021625    483497 13-APR-94
00001-678901234567890123456789012345678

       10         1     54990    437546 13-APR-94
00001-678901234567890123456789012345678

    15401      1541  93221027    478443 13-APR-94
00001-678901234567890123456789012345678

      701        71   7034285    386957 13-APR-94
00001-678901234567890123456789012345678

       36         4    318489     96450 13-APR-94
00001-678901234567890123456789012345678

      751        76   7531070    320344 13-APR-94
00001-678901234567890123456789012345678

    15405      1541 154051500    -62465 13-APR-94
00001-678901234567890123456789012345678


14 rows selected.

SQL> exit
```

Appendix C: Tunable Parameters

Challenge XL Operating System Tunable Parameters

```
rlimit_rss_cur          = 0x20000000
maxlkmem                = 0x100000 (or is it 23593 = 0x5c29?)
posix_tty_default       = 1
resettable_clocal       = 1
nproc                   = 500
nprofile                = 200
maxup                   = 400
semmni                  = 100
semmns                  = 200
semmnu                  = 100
semmsl                  = 200
semopm                  = 200
ndpri_hilim             = 30
```

Indy Operating System Tunable Parameters

```
rlimit_rss_cur  = 40894464
rlimit_nofile_cur = 500
nproc                   = 1900
nprofile                = 200
maxup                   = 1900
semmni                  = 1300
semmns                  = 1300
semmnu                  = 1300
semmsl                  = 200
semopm                  = 200

sshmseg                 = 150

msgmni                  = 1400
msgtql                  = 2800
msgseg                  = 16384
msgmnb    = 380000
msgssz    = 100
```

ORACLE7 Configuration

Init.ora Parameters

```
#
# $Header: /oradev/bench/tpc/tpcab/admin/RCS/p_run.ora,v 1.8 1993/08/21
21:36:46 oradev Exp oradev $ Copyr (c) 1993 Oracle
#
db_name   =  tpcb
db_files= 150
max_rollback_segments   =160
db_file_multiblock_read_count = 8
db_block_checkpoint_batch =128
```

```
dml_locks=800
log_archive_start=FALSE
log_checkpoint_interval =999999999999
log_checkpoints_to_alert = TRUE
log_buffer=327680
processes=300
transactions_per_rollback_segment = 1
rollback_segments       = (s1, s2, s3, s4, s5, s6, s7, s8, s9, s10,
                           s11, s12, s13, s14, s15, s16, s17, s18, s19,
s20,
                           s21, s22, s23, s24, s25, s26, s27
             )
sessions= 400
single_process          =FALSE
log_archive_start       =FALSE
discrete_transactions_enabled = TRUE
cursor_space_for_time=TRUE
shared_pool_size =7000000

# use_post_wait_driver
use_post_wait_driver=TRUE
post_wait_device=/dev/postwait
use_async_io=TRUE

db_block_buffers = 37000
_db_block_write_batch = 526

spin_count = 6000

#
sort_area_size=  52428800
```

Appendix D: Storage Requirements

Disk Storage Requirements

According to Clause 9.2.4.1 the priced configuration must contain sufficient disk space to store 8 hours of log data and 90 days of history data. This section documents the disk storage requirements of the priced configuration.

The total disk space requirements can be calculated as follows:

IRIX (UNIX system files + system swap space) +
ORACLE system and control files +
ORACLE database (ABT) files +
ORACLE log data (8-hours, mirrored) +
History data (90 days)

The 8-hour log data requirement was determined as follows:

At 2049.71 TPS-A, the checkpoint interval was 2329 seconds and the log size was 2000 Mbytes.

For 8-hour, the un-mirrored log requirements are
8 hours * 3600 seconds/hours * 2000 Mbytes / 2329 seconds =
24731.65 Mb
The mirrored log requirements are
2 * 24731.65 = 49463.30 Mb

Disk drive capacity = 1918.2 Mb (formatted)

A total of 26 disk drives were dedicated to the 8-hour mirrored log rquirement with space allocated as follows:

Total space available = 26 drives * 1918.2 Mb/drive = 49873.2 Mb

Log space required = 49463.30 Mb

Unused capacity = 49873.2 - 49463.3 = 409.9 Mb

The size of a history file entry was determined as follows:

The 90-day history file space requirements was computed as follows:

Average number of history file entries per 2K page = 26.24
Number of daily transactions = 8 hours * 3600 * 2049.71 = 59,031,648
Number of transactions for 90 days = 90 * 59,031,648
= 5,312,848,320

Number of 2K blocks required = 5,312,848,320 / 26.24 = 202471353.66
= 395451.86 Mb

Total disk space available for the 90-day history file requirement was computed as follows:

Disk space utilization:

UNIX system files	534.6 Mb
Swap space	256.0 Mb
ORACLE system and control files	1738.2 Mb
Branch and Teller files	900.0 Mb
Account files	53760.0 Mb
Account index	5050.0 Mb

Total (less log and history) 62238.8 Mb

Total number of drives (less log files) 239
Total space available (less log space) 239 drives * 1918.2 Mb =
458449.8 Mb

Total space available for 90-day history file requirement =
458449.8 - 62238.8 = 396211.0 Mb

Total number of drives including log files 239 + 26 = 265 drives

Appendix E: Attestation Letter

PERFORMANCE METRICS INC.
TPC Certified Auditors

Mr. Carl Rigg
Manager, Software Development
Interactive Products Division
Silicon Graphics, Inc.
2011 N. Shoreline Blvd.Mail Stop 8U-500
Mt. View, CA 94039-7311

April 17, 1994

I remotely verified the performance of Oracle7 performing the TPC Benchmark™ A on the CHALLENGE XL Server. The operating system was IRIX 5.3 (UNIX). The results were:

Model	CPU's	Memory	Disks Measured	Response Time @ 90%	tpsA
Server					
CHALLENGE XL	32 MIPS R4400 @ 150 Mhz	Main: 1024 MB 16K icache 16K dcache 4M secondary	265 @ 1.92 GB each	<1.6 seconds	2049.71
Client Configuration					
20 Indy Workstations	R4000SC @ 100 Mhz	256 MB	1 GB	n.a.	n.a.

The following attributes of the benchmark were remotely verified:

- The transaction was correctly implemented
- The application used a single history table
- The database records were the proper size
- The database was properly scaled
- Sufficient mirrored log data was configured
- The ACID properties were met
- The cycle time of 10.167 exceeds the cycle time of theSpecialix test
- The response time includes 0.58 seconds for Speicalix and RS232 delays
- The steady state portion of the test was 36 minutes, 40 seconds
- One checkpoint was taken during the steady state portion of the test
- One log switch occurred during the steady state portion of the test
- A price sheet was checked for components and maintenance

Respectfully yours,

Lorna Livingtree

Lorna Livingtree
Auditor

Appendix F: RTE

RTE

rte.c

```
/*=================================================================+
 |          Copyright (c) 1992  Oracle Corp, Belmont, CA         |
 |                    UNIX PERFORMANCE GROUP                    |
 |                     All Rights Reserved                      |
 +=================================================================+
 | FILENAME
 |    drive_a.c
 | DESCRIPTION
 |    confidential TPC-A Remote Terminal Emulator process
 +=================================================================*/

              .
              .
              .

  trunintegral = (int) (11 * thinktime); /* truncate think time */

  /* initialize timing */

  begin_time = starttime + (double) ramp_up;
  end_stat_time = begin_time + (double) timelimit;
  end_time = end_stat_time + (double) ramp_down;

  /* execute transaction until time is up */

  while (1)
  {
    /* think */
    integral = (int) ceil(-thinktime *
                 log(1.0 - ((lrand48() % 32768) / (float) 32768)));

    if (integral > trunintegral)
      integral = trunintegral;
```

```
if ( integral < (termdelay * 1000) )
     integral = (int) (termdelay * 1000);

poll (0, 0, integral);

/* generate input data */

delta = (lrand48 () % 1999999) - 999999;

if (lrand48 () % 100 < 85)
    account_branch = branch;
else
{
    account_branch = (lrand48 () % mult) + 1;
    if (account_branch >= branch)
        account_branch++;
}

account = (account_branch - 1) * 100000 + (lrand48 () % 100000) + 1;

/* get start time */

tr_begin = gettime ();
if (in_ramp_up && (tr_begin > begin_time))
{
    in_ramp_up = 0;
    in_timing_int = 1;
}

/* execute transaction */

              .
              .
              .

tr_end = gettime () + termdelay;
tr_time = tr_end - tr_begin - tr_overhead ;
if (in_timing_int)
{
    if (tr_end < end_stat_time)
    {
        tr_count++;
        if (account_branch != branch)
            remote++;
        if (tr_time <= FAST_LIMIT)
            tr_fast++;
        if (tr_time < tr_min)
            tr_min = tr_time;
        if (tr_time > tr_max)
            tr_max = tr_time;
        tr_sum += tr_time;
        tk_time = integral / 1000.0;
        if (tk_time < tk_min)
            tk_min = tk_time;
        if (tk_time > tk_max)
            tk_max = tk_time;
        tk_sum += tk_time;
        if ((i = tr_time / BUCKINT) >= NBUCK)
        i = NBUCK - 1;
        timing_buckets[i]++;
        if ((i = tk_time / TBUCKINT) >= NTBUCK)
        i = NTBUCK - 1;
        think_buckets[i]++;
    }

                    .
                    .
                    .
```

Appendix G: Third-Party Vendor Quotations

The following pages are price quotes from third-party vendors for this benchmark.

Specialix International
Worldwide OEM Group

Specialix

March 9, 1994

745 CAMDEN AVENUE
CAMPBELL, CA 95008
TELEPHONE: (800) 423-5364
(408) 378-7919
FAX: (408) 378-0786

Silicon Graphics
2011 N. Shoreline Blvd.
Mountain View, CA 94039

TPC Price Schedule

Quality	MTS	MTA	5 yr. Warranty
0-25	1017	420	$25.00
25-100	932	357	$25.00
100-500	847	325	$25.00
500-1000	678	305	$25.00
~~1000-1500~~	~~678~~	~~286~~	$25.00
1500 +	678	228	$25.00

Note

Pricing used in Silicon Graphics TPC test based on quantity of 625 MTS and 1875 MTA to configure the 20000 ports.

Pricing is based on Single bulk shipment of a minimum of the applicable quantity level.

Site Software license in included at no charge.

Pricing is available only directly from Specialix, Inc.
745 Camden Avenue #A
Campbell, CA 95008-4146
Attn: Dave Stege
800-423-5364

Terms:	Net 30 Days
FOB:	Campbell, CA.
Delivery:	2 weeks ARO

Regards,

Dave Stege
Director of USA OEM Sales

WYSE TECHNOLOGY
TERMINAL PRODUCT PROPOSAL
APRIL 14, 1994

1. Customer: Silicon Graphics (SGI)

2. Customer Requirements: Entry-level, ASCII terminal and keyboard

3. Volume: 20,000 units

4. Wyse Product: WY-30+

5. Warranty: Standard 1 year warranty included in quoted price
Optional 5 year warranty added at $35.00 per terminal. This 5 year warranty is inclusive of the 1 year standard warranty bundled with the terminal

6. Unit Pricing:

Quantity	Price Per Unit
5,000 - 9,999	$203.00
10,000 - 19,999	$196.00
20,000 - 24,999	$189.00

7. Terms and Conditions:

- One time purchase of volume quantity
- Minimum order quantity of 45' foot container load @ approximately 520 units per container
- Terminals must be sold in conjunction with an SGI product(s) and must not be resold separately into the distributor channel
- FOB Taiwan
- Quote is for Wyse standard product and does not include customization of any sort. If customization is required, Wyse will provide a new quote.
- This quote is valid for 90 days from date quoted.

Approved: Terry Eastham Date: 4/14/94
Terry Eastham, VP, Displays Marketing

ANIXTER®

QUOTATION

See Reverse for Conditions of Sale

Date 04/13/1994	Quote No. 00143
Customer No. 529610	

Farokh Mehr
Silicon Graphics

Anixter Inc.
Corporate Headquarters
4711 Golf Road
Skokie, Illinois 60076

Item	Quantity	Anixter Catalog Number and Description	Unit	Unit Price	Extended Price
01	6	148592 MICRO HUB 1E 9 PORT 10BASET WORKGROUP HUB product has 5 year warranty price good for 90 days	EA	375.00	2250.00
02	22	148617 MICRO MAU BNC TO AUI THINNET TRANSCEIVER product as 5 year warranty price good for 90 days	EA	49.00	1078.00
		Page Total:			3328.00
		Quote Total:			3328.00

TERMS NET30
F.O.B. SHIP.PT., PPD/CHARGE
SHIPMENT Orders shipped UPS unless otherwise specified.
NOTE Order acceptance based upon credit approval by Anixter.

Prices will be those in effect at time of shipping.

Please refer all inquiries to:

Beth May
Anixter Inc.
2151 O'Toole Ave Suite H
San Jose, California 95131
Phone: (408) 435-1212
Fax: (408) 435-9174

Beth May